Rockwell International

Rocketdyne Division
6633 Canoga Avenue
Canoga Park, California 91304

RI/RD84-191

RESEARCH STUDY FOR EFFECTS OF CASE FLEXIBILITY
ON BEARING LOADS AND ROTOR STABILITY
FINAL REPORT

31 August 1984

Contract NAS8-34964

31 August 1984

PREPARED FOR
NASA GEORGE C. MARSHALL SPACE FLIGHT CENTER
HUNTSVILLE, ALABAMA

PREPARED BY

J. R. Fenwick R. B. Tarn

J. R. FENWICK/R. B. TARN

APPROVED BY

G. F. Bremer

G. F. Bremer
PROJECT MANAGER

Page intentionally left blank

Page intentionally left blank

FOREWORD

This report was prepared in compliance with Contract NAS8-34964 under G.O. 945301.

ABSTRACT

Methods to evaluate the effect of casing flexibility on rotor stability and component loads are developed. A survey of some more recent Rocketdyne turbomachinery was made to determine typical properties and frequencies versus running speed. A small generic rotor was run with a flexible case with parametric variations in casing properties for comparison with a rotor attached to rigid supports. A program was developed for the IBM-Personal Computer for interactive evaluation of rotors and casings. The Root Locus method was extended for use in rotor dynamics for symmetrical systems by transforming all motion and coupling into a single plane and using a 90 degree criterion when plotting loci.

CONTENTS

ILLUSTRATIONS

TABLES

1.0 INTRODUCTION

High-speed, high-performance turbomachinery is as light in weight as possible. A minimum of weight is added for the specific purpose of stiffness whether in the rotor, casing or mount. The major purpose of this study was, therefore, to develop tools to evaluate the effects of casing motion on the stability of such high speed rotors.

Several approaches were used in this evaluation. First, a survey of typical properties for the most modern pumps made at Rocketdyne was made. Second, a brief study was made of a fairly simple rotor and casing which is generally similar to the SSME main pumps. A third effort involved evaluating several existing mainframe computer programs and adapting them into a single program which could be run on a minicomputer (IBM-PC) in an interactive mode. The fourth aspect of the study was to determine if the use of an entirely different approach, the root locus, could be adapted for use in rotor dynamics to achieve additional insight to the coupling and stability problems of turbomachinery with flexible casings.

2.0 SUMMARY

2.1 HIGH SPEED PUMP SURVEY

The physical properties of a number of existing high speed rotors was accumulated and summarized. This summary is included in Appendix A. As an aujunct to this survey, existing models for many of the machines were also reviewed to determine the extent of models used to evaluate stability and loads. It was rather interesting that initial models using a rigid mount for the bearings appeared to produce a pessimistic stable speed range. In later stages in pump development when relatively complex casing models were added, the stable speed range was computed to be higher. Two possible explainations were suggested. The first was that casing motion itself may increase the stability range. The second was that an increase in flexibility of the case in one plane might act like an asymmetric bearing stiffness. This has been shown to be a stabilizing factor by several studying this problem.

An additional bit of information found in the survey and review was that in one study, modal truncation had been used to evaluate its effect on a pump model. In this particular case, reasonable results were obtained when only modes through the first rotor flex mode were retained.

This indicated that perhaps a very simple rotor model would be sufficient for generic coupling studies.

2.2 GENERIC CASING MOTION MODEL

Analysis of six turbopumps demonstrates that asymmetric housing flexibility increases the rotor stability. A rigid housing assumption predicts an instability to occur at lower speed than an analysis with a flexible housing. Consequently, a rigid housing assumption is more conservative than a flexible housing model. The flexible housing acts as en energy absorber and dissipates some of the destabilizing forces in the rotor/housing system.

A simple 20-lumped-mass turbopump model (10 mass rotor, 10 mass casing) yields consistent results with the more complex, detailed turbopump models. That is, the rigid housing again renders conservative results. However, it is necessary to comply with certain prescribed conditions which are very typical of turbopumps. It was determined that the asymmetric spring stiffness ratio of the supports grounding the housing should be greater than or equal to 3. In addition, there is an optimum value of the asymmetric spring stiffness ratio of approximately 5. It is not fully understood why stability decreases with asymmetry greater than a 5:1 ratio. Another recommended condition is that the housing-to-rotor weight be at least 6:1 based on this weight ratio being representative of many existing turbopumps. Adherence to these conditions ensures that the simple 20-lumped-mass turbopump model will yield consistent results with the more detailed models.

2.3 PC ROTOR-CASING MODEL

A full description of the Rotor-Casing model is included. The program is the result of compacting a number of large existing codes written for CDC and IBM mainframe computers. The language used throughout the program is Microsoft-FORTRAN and is compatible with IBM-PC, DOS-1.1 and subsequent operating systems. Although the program is capable of handling many degrees of freedom, for interactive use it is recommended that simpler systems be used to minimize computing time and maximize interaction.

A feature of the output summary is a plot of the locus of roots as speed is varied. Reference lines equivalent to various damping (percents of critical) are included. As speed is increased, the small changes in modal frequency (due to gyroscopics) and the change in damping (angle in the polar plot) are displayed quite vividly. The transition of a root into the unstable right hand plane is easily seen as is the migration of roots which remain stable.

A very significant feature is that a root which becomes unstable can easily be traced back to its low-speed location. Since the cross coupling affects damping with negligible change in mode shape, sensitive system mode shapes are easily recognized.

2.4 THE ROOT LOCUS METHOD

Initially a Jeffcott rotor was examined to see if this approach had merit in rotor dynamics. The result of this application was quite promising. Basically the approach uses the dynamics of the system at zero speed as a baseline for dynamics. As speed is increased, both direct planar coupling coefficients and cross coupling coefficients increase. The simple model showed that a meaningful plot was the locus of system characteristic roots with speed as a parameter.

Most interesting was that the basic system dynamics could be evaluated in one plane (X, Z) and that cross coupled forces could be transformed back into plane by recognizing in-plane forces (F (X)) result in orthogonal motion (ΔY) but that for a symmetrical system $\Delta Y = i\Delta X$. Thus, only a planar model should be necessary if root locus can be adapted for multiple feedback loops using a 90-degree angle criterion.

The problem was approached from a number of directions. Basically the simplest model must include a flexible casing with mass supported on two springs. Two bearings connect the rotor and casing, gyroscopic cross coupling occurs at the impeller and turbine, while cross coupled stiffness occurs between the rotor masses and the casing. A shaft with an overhung turbine appeared to be the most general case and was used exclusively. Deleting the turbine as an item and adding its properties to the impeller would serve to estimate the dynamics of a pump with outboard bearings.

Despite several attempts to generate closed form solutions which could be used with a root locus approach, the problem soon became too unwieldy. One attempt, based on the Microsoft version of µ-math, was successful in showing the complexity and futility of this direct approach.

A less direct approach was used in which the planar model was described in terms of matrices based on primary properties of the case and rotor. Modal transfer functions at the two cross coupled stiffness locations and at the two rotor masses were developed. A method for closing the cross coupling loops one by one (first gyroscopic then hydrodynamic) was generated. A root locus approach using a general angle criterion was also generated.

A program was written in Microsoft-FORTRAN to evaluate the locus of roots for a single primative mode at a time. Major problems were encountered in implementing the loop closure section of the program. Some bugs were suspected in the double precision implementation of the complex arithmetic statements of the issue of FORTRAN. A later version is being obtained but has not yet been received. The formulation of the loop closure techniques is also being reviewed.

At this time the only tool available for the case housing studies is the one previously discussed. A root locus display is included and an indication of the sensitivity of stability to speed is readily seen.

3.0 REVIEW OF ROTORS AND CASINGS FOR ROCKETDYNE TURBOPUMPS

Table 1 lists pumps which have been built by Rocketdyne. Many of the earlier pumps were driven through a gear box and were not capable of speeds where casing flexibility would play a role in rotor dynamics. Some used both a fuel and oxidizer impeller on a common drive shaft, which normally results in a relatively low-speed drive and a heavy, stiff casing. Some of the experimental pumps were designed with strong heavy rather than flight weight casings since details of hydrodynamics were being studied. In summary, a flexible casing model was never required to support many of the pump designs since the casing did not affect rotordynamics.

Data for several recent pumps are contained in Table 2. The approach here is to summarize pertinent design information for the pumps and to list the mass properties for the casing and rotor as well as free-free frequencies for the rotor and frequencies for the casing attached to its mount points.

TABLE 1. ROCKETDYNE PUMPS

Model	Pump Type	Application	Fluid	Flow, gpm	Discharge Pressure, psia	Speed		Units Produced
						rpm	hp	
MK-2	Single-Stage Centrigual	Redstone	Ethel Alcohol	1268	450	4840	836	155
			LOX	1294	340	4840		155
MK-3	Single-Stage Centrifugal	Jupiter, Atlas Thor, H-1	RP-1	2000	950	31,600	3550	2551
			LOX	3210	800	31,600		
MK-4	Single-Stage Centrifugal	Atlas	RP-1	730	900	10,000	1540	712
			LOX	1180	900	10,000		
MK-5	Barske	AR	RP-1	25	560	25,000	50	30
			H_2O_2	107	860	25,000		
MK-6	Single-Stage Centrifugal	E-1	RP-1	3200	1300	7900	10,800	2
			LOX	5100	1100	7900		
MK-9	Seven-Stage Axial	Nuclear Feed	LH_2	10,600	1500	33,900	15,000	10
----	Piston	LH_2 Micropump	LH_2	.06.3	2300	1200		1
MK-10	Single-Stage Centrifugal	F-1	RP-1	15,500	1960	5600	59,350	160
			LOX	24,640	1550	5600		
MK-14	Centrifugal Mixed Flow	H-2	RP-1	2670	1260	14,300	7650	3
			LOX	2340	1175	14,300		
MK-15	Six-Stage Axial	J-2	LH_2	7960	1050	26,050	5940	210
	Centrifugal	J-2	LOX	2840	920	8650	2340	200
MK-19	Single-Stage Centrifugal	X-8	LH_2	3770	1010	27,000	4100	2
			LOX	1160	680	8750	760	2
MK-22	Multi-Stage Radial	Experimental (MK-22)	LH_2	4000	5400	38,000	17,000	3
MK-25	Four-Stage Axial	Nuclear Feed	LH_2	18,500	2250	34,000	21,000	8
MK-29	Two-Stage Centrifugal	J-2S	LH_2	10,200	2000	30,500	14,800	10
MK-29	Single-Stage Centrifugal	J-2S	LOX	3270	1520	10,350	3850	10
MK-35		Nuclear Feed	LH_2	6100	1600	34,000	7300	2
MK-36	Centrifugal	F_2	F_2	12	1500	72,000	22	2
MK-37	Gear	F_2	F_2	12	1500	5200	15	2
MK-44	Two-Stage Centrifugal	APS	LH_2	4600	1600	60,000	800	2
	Single-Stage Centrifugal	APS	LOX	100	1600	30,000	150	2
----	----	Preinducer	LOX	2300	90	3930	170	1
MK-39	Three-Stage	SSME LPFTP	LH_2	16,320	260	15,800	2950	18
MK-38	Centrifugal	SSME HPFTP	LH_2	16,360	7000	37,355	76,560	19
MK-39	Two Stage	SSME LPOTP	LOX	6080	440	5450	1740	19
MK-38	Centrifugal	SSME HPOTP	LOX	7240	8435	31,100	27,770	17
MK-48	Three-Stage Centrifugal	ASE	Fuel	628	4560	95,000	2543	2
	Single-Stage Centrifugal	ASE	LOX	225	4320	70,000	856	2

TABLE 2. TURBOPUMP DYNAMIC SUMMARY

DESIGN PARAMETERS						
MODEL	PRESSURE, PSI	FLOW, GPM	POWER, HP	SPEED, RPM	ROTOR WEIGHT, POUNDS	CASE WEIGHT, POUNDS
MK-38 (CH_2)	7000	16,360	76,560	37,335	128.5	649
MK-38 (O_2)	4800	7,249	27,770	31,100	81.0	480
MK-48 (CH_2)	4560	628	2,543	95,000	7.3	48
MK-49 (CH_2)	4918	4.2	1,704	110,000	6.1	48

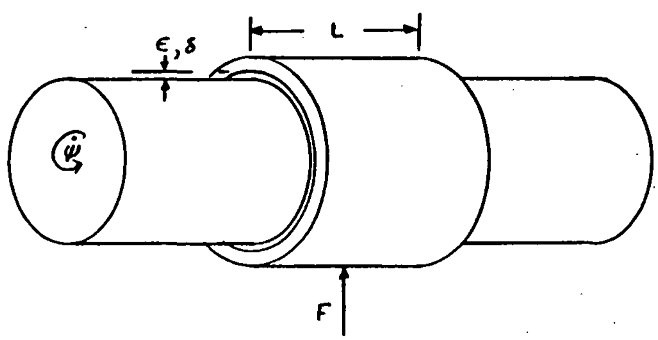
COMPONENT FREQUENCIES (HZ)							
MK-38 (CH_2)		MK-38 (O_2)		MK-48 (CH_2)		MK-49 (CH_2)	
ROTOR	CASING	ROTOR	CASING	ROTOR	CASING	ROTOR	CASING
643	51	429	45 351	683	247	1050	36
1319	116	900	73 375	1721	336	2528	611
	135		86 432		537		1544
	387		112 468		798		1631
	424		134 488		906		
	725		171 542		1010		
			188 571		1339		
			231 587		1400		
			278 613		1418		
			300 650		1621		
			310 705		1747		
					1769		
					2313		

4.0 COMPONENT MODELS

4.1 BEARINGS

The primary reaction of roller and ball bearings is that of a spring. In some designs there is a small fluid-filled annular gap between the outer race and the bearing carrier to allow axial shaft motion and casing growth. For purposes of stability and bearing load, the annular gap is generally ignored because even very small radial shaft motion closes the gap. Until the gap is closed, the bearing loads are small.

Hydrodynamic bearings (journal bearings) do not have a direct stiffness component. As shown in Fig. 1 the force has off diagonal terms: e.g., forces developed in one direction due to displacement in the other axis.



$$\begin{Bmatrix} F_Y \\ F_Z \end{Bmatrix} = \frac{K\dot{\psi}}{2}\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}\begin{Bmatrix} \delta_Y \\ \delta_Z \end{Bmatrix} + K\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{Bmatrix} \dot{\delta}_Y \\ \dot{\delta}_Z \end{Bmatrix}$$

$$K = 2\mu R L^3 \pi / \epsilon^3$$

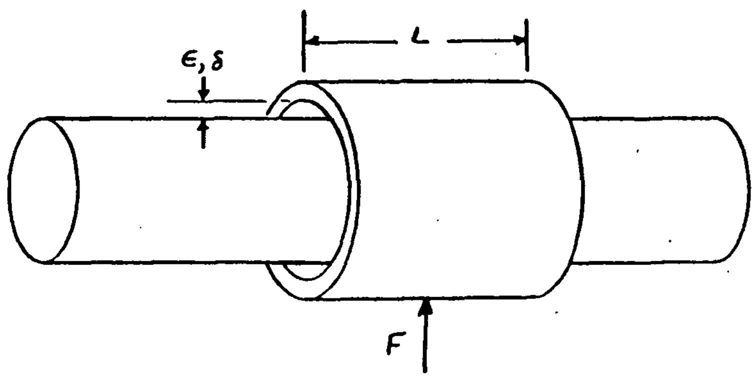
Figure 1. Journal Bearing

Hydrostatic bearings are somewhat similar to journal bearings, except that recessed pockets are designed into the nonrotating part. Fluid is fed through an orifice associated with each pocket. A pressure drop is also taken across the pocket land rotor clearance. Shaft radial motion causes an increase in pocket pressure, which generates a direct stiffness component. In most high-performance pumps, the pressure source for the bearing is the discharge pressure of the pump so that the stiffness is a function of speed. The lands themselves may contribute forces similar to a hydrostatic bearing.

4.2 CONTROLLED LEAKAGE SEALS

Labyrinth and smooth annular seals can act as bearings contributing stiffness and damping, which carry forces between the casing and rotor due to relative deflection. The grooves of a labyrinth tend to distribute pressure circumferentially, but the teeth operate as a series of individual straight smooth seals.

A straight smooth seal is shown schematically in Fig. 2. The dynamic properties of the seal are written in coordinates of the fluid which rotate at half the rotor speed. These can be transferred to the nonrotating coordinates of the casing as shown. By changing the signs of off-diagonal terms, the equations represent the equation in coordinates rotating with the shaft. The diagonal terms are passive or damping components, while the off-diagonal terms can lead to instability.



HALF SPEED COORDINATES

$$F = K_o \delta + D_o \dot{\delta} + M_o \ddot{\delta}$$

$$K_o = \pi R L Q \sigma / \epsilon (1 + 4\sigma/3)$$

$$D_o = \pi R L^2 Q (1 + 2\sigma/3)^2 / \epsilon V (1 + 4\sigma/3)$$

$$M_o = \pi R L^3 \rho / 12 \epsilon (1 + L^2/3D^2)$$

FIXED COORDINATES

$$\begin{Bmatrix} F_Y \\ F_Z \end{Bmatrix} = \begin{bmatrix} K_o - M_o(\dot{\psi}/2)^2 & D_o \dot{\psi}/2 \\ -D_o \dot{\psi}/2 & K_o - M_o(\dot{\psi}/2)^2 \end{bmatrix} \begin{Bmatrix} \delta_Y \\ \delta_Z \end{Bmatrix} + \begin{bmatrix} D_o & M_o \dot{\psi} \\ -M_o \dot{\psi} & D_o \end{bmatrix} \begin{Bmatrix} \dot{\delta}_Y \\ \dot{\delta}_Z \end{Bmatrix} + \begin{bmatrix} M_o & 0 \\ 0 & M_o \end{bmatrix} \begin{Bmatrix} \ddot{\delta}_Y \\ \ddot{\delta}_Z \end{Bmatrix}$$

Figure 2. Smooth Straight Seal

4.3 TURBINE BLADE FORCES

Turbine disk (Alford) forces are generated by changes in radial clearance of the turbine tips. These forces are generally written as

$$\begin{Bmatrix} f_y \\ f_z \end{Bmatrix} = \begin{bmatrix} 0 & -k \\ k & 0 \end{bmatrix} \begin{Bmatrix} \delta_y \\ \delta_z \end{Bmatrix}$$

$$K = \frac{\beta x \ h \ x \ 31{,}500}{R \ x \ h \ x \ N}$$

$$1 < \beta > 2$$

While this relationship is not as clearly documented as some other effects, it difinitely exists and due to its off-diagonal form, is a destabilizing whirl driver. Since all the power to the pump is delivered by the turbine, only small amounts of power oriented at 90 degrees to the deflection can result in large rotor vibration. It must be absorbed by natural damping contained in nonrotating elements.

5.0 EFFECTS OF HOUSING MOTION ON ROTOR STABILITY

A rotor stability analysis has been performed for the six turbopumps specified in Table 3. These turbopumps were selected because housing models currently exist and, therefore, could be analyzed for both a rigid and a flexible housing assumption to determine the effect on rotor stability. Figure 3 through 10 present a stability map for each turbopump where the real part of the complex eigenvalue is plotted as a function of the rotor spin speed. Eigenvalues with a positive real part indicate the potential for unstable rotor motion. Stability maps for the second rotor modes were excluded for cases where the second rotor mode is stable. Observing Fig. 3 through 10, as the rotor spin speed increases, the rotor in a flexible housing tends to approach an unstable condition subsequent to the rotor in a rigid housing. Thus, the rotor with a rigid housing predicts an instability prior to the rotor with a flexible housing assumption.

TABLE 3. TURBOPUMP DESIGN PARAMETER SUMMARY

MODEL	PRESSURE, PSI	FLOW, GPM	POWER, HP	SPEED, RPM	ROTOR WEIGHT, POUNDS	CASING WEIGHT, POUNDS	BEARING SPAN, INCHES
MK-380 (104%)	4312, 7442	7093, 707.6	25357	28057	81	480	10.89
MK-380 (109%)	4555, 7861	7404, 774.8	29173	29374	81	480	10.89
MK-380 (REDESIGN)	4800, 8000	7240, 652	29000	26030	112	480	13.40
MK-38F	7000	16360	76560	37335	128.5	649	23.50
MK-48F	4560	628	2543	95000	8.25	87.75	7.74
MK-49F	4918	418.3	1704	110000	6.1	48	7.36

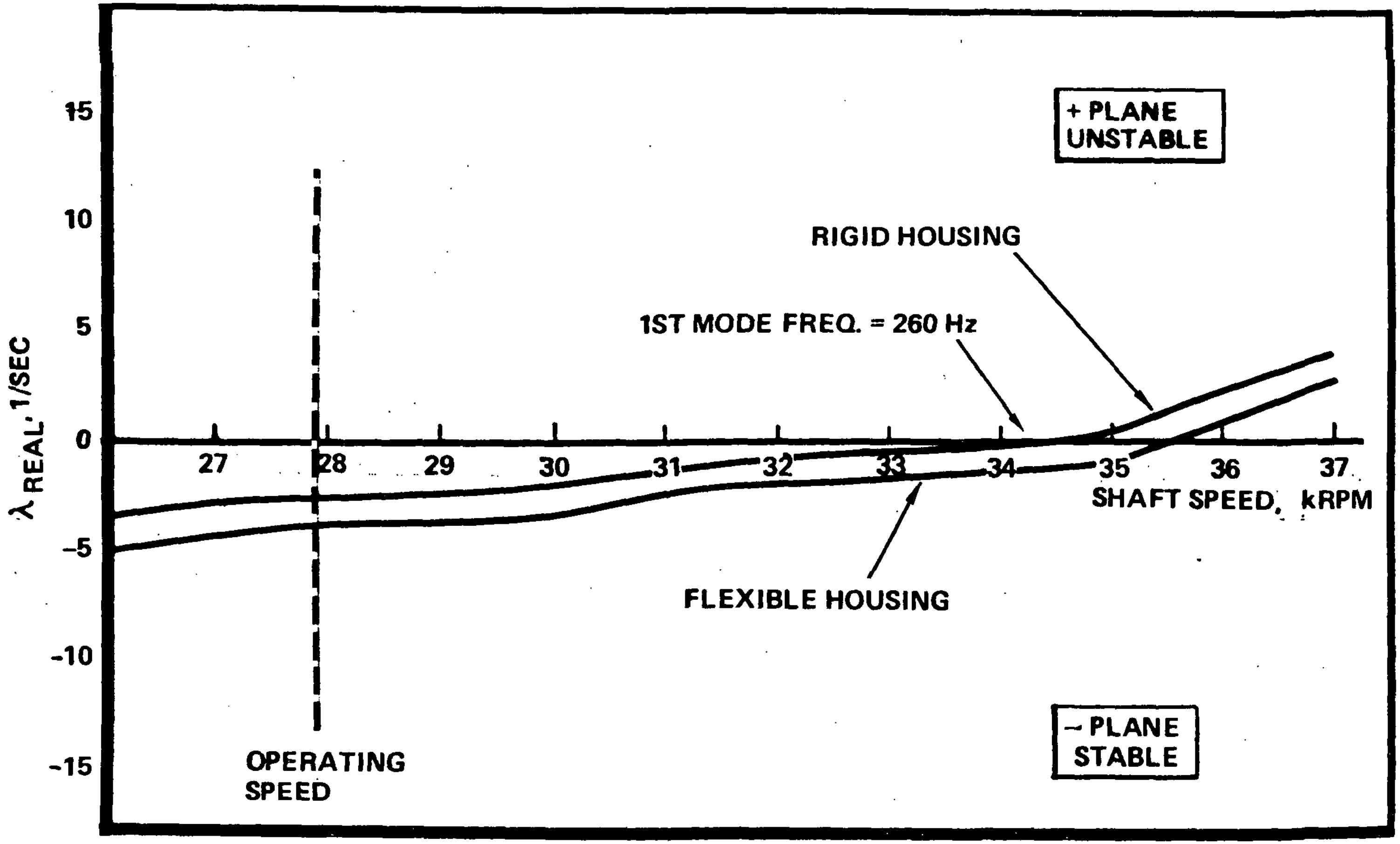


Figure 3. MK-380 (104%) Stability Map of First Rotor Mode

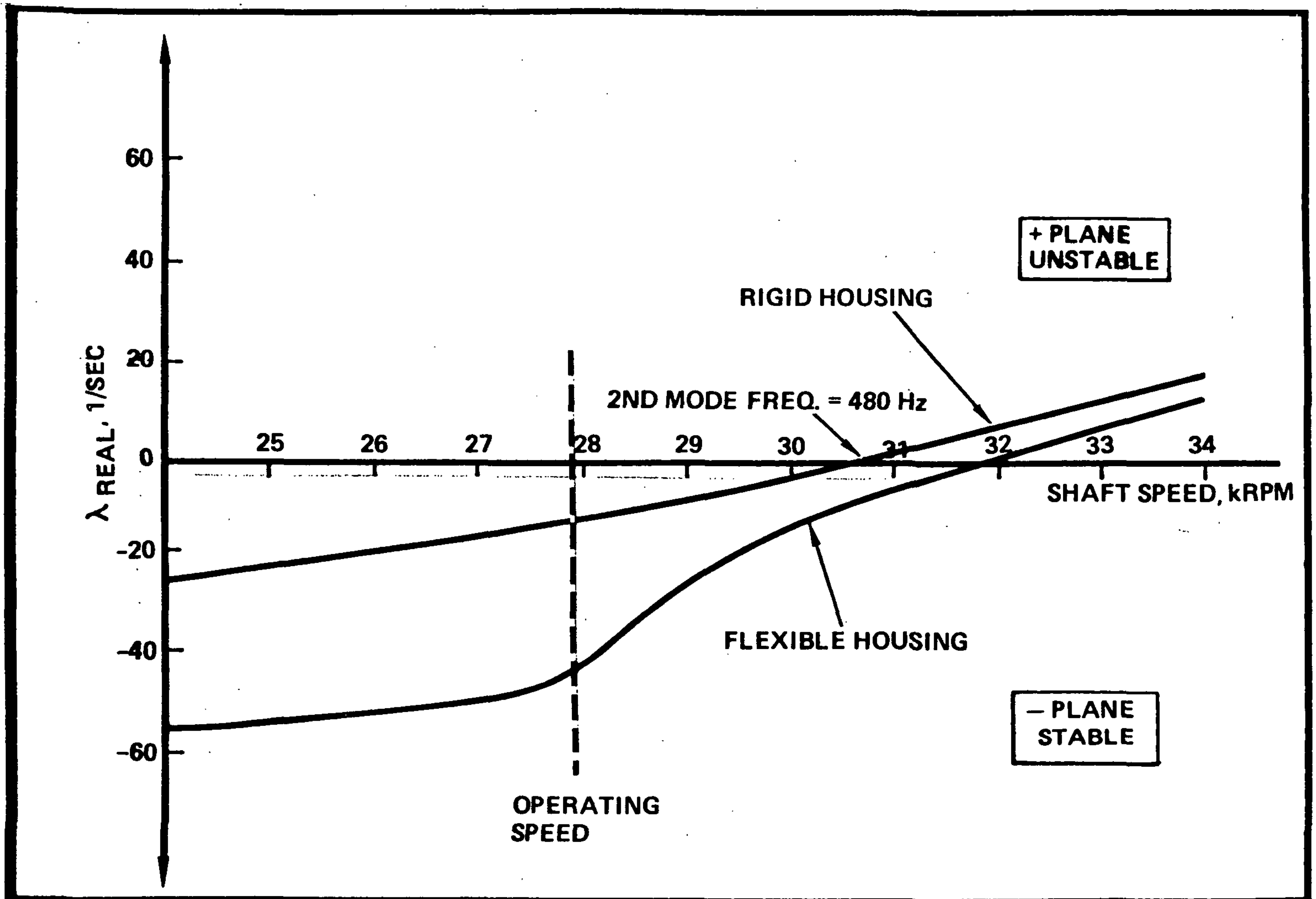


Figure 4. MK-380 (104%) Stability Analysis of Second Rotor Mode

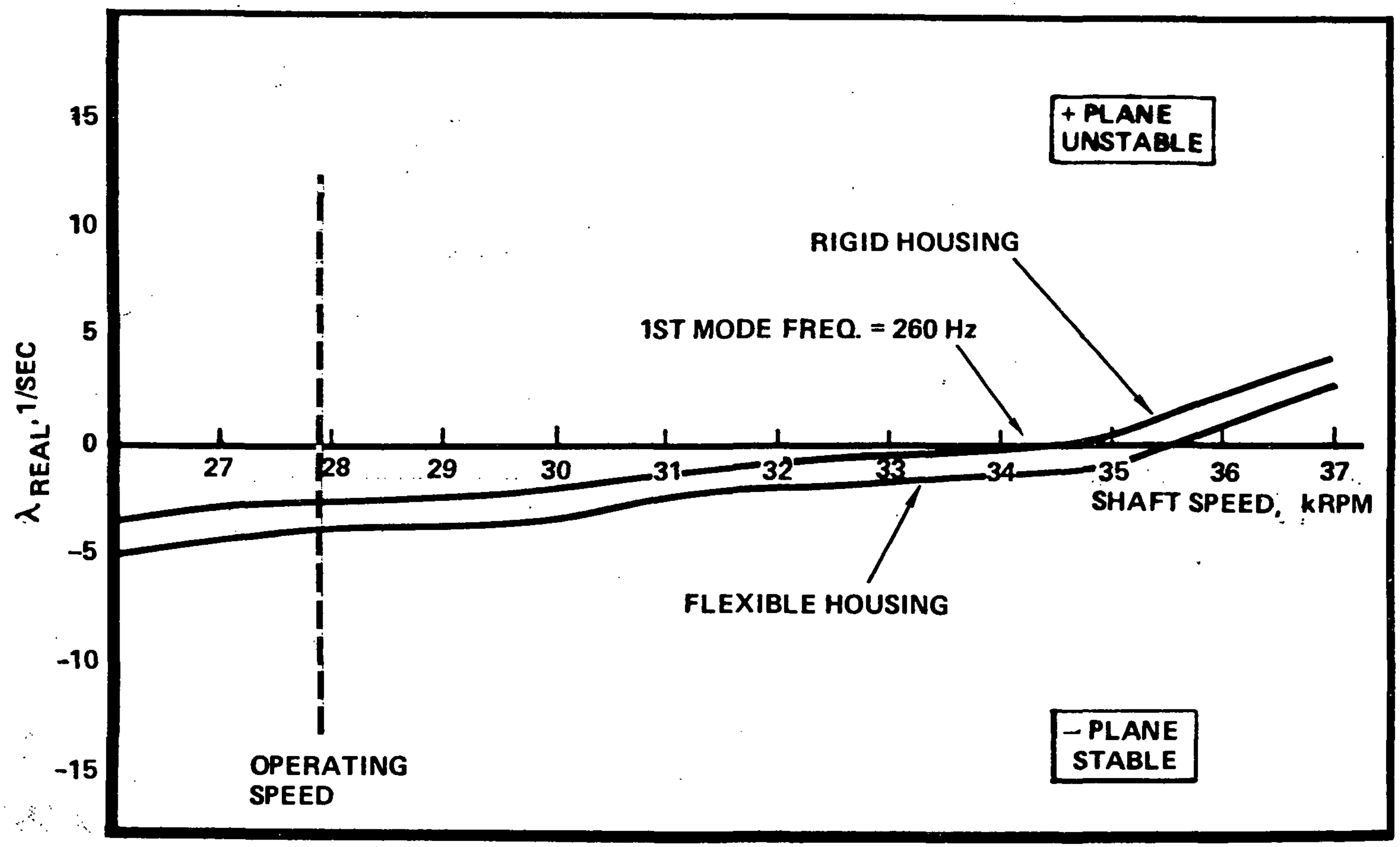


Figure 5. MK-380 (109%) Stability Map of First Rotor Mode

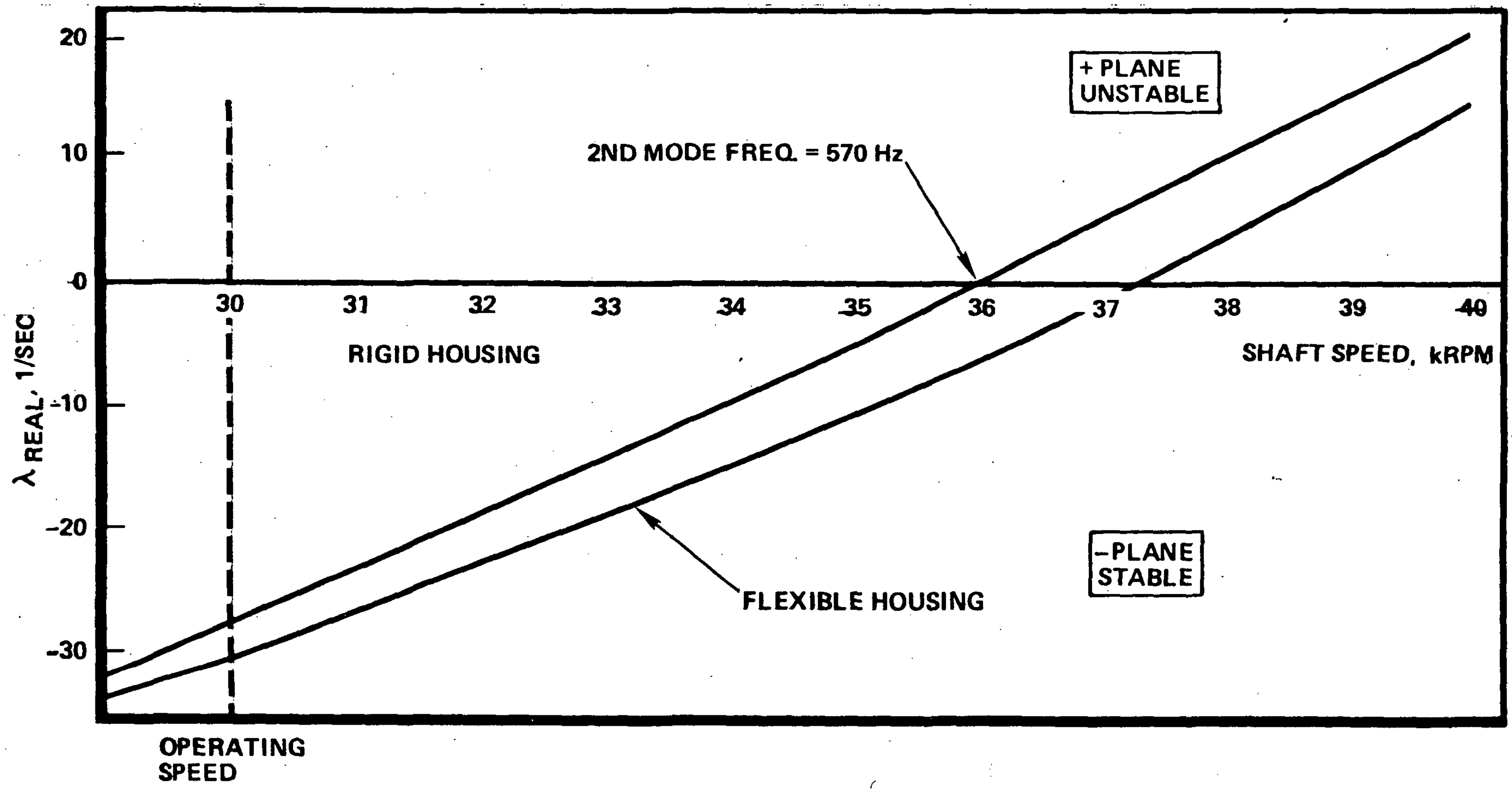


Figure 6. MK-380 (109%) Stability Map of Second Rotor Mode

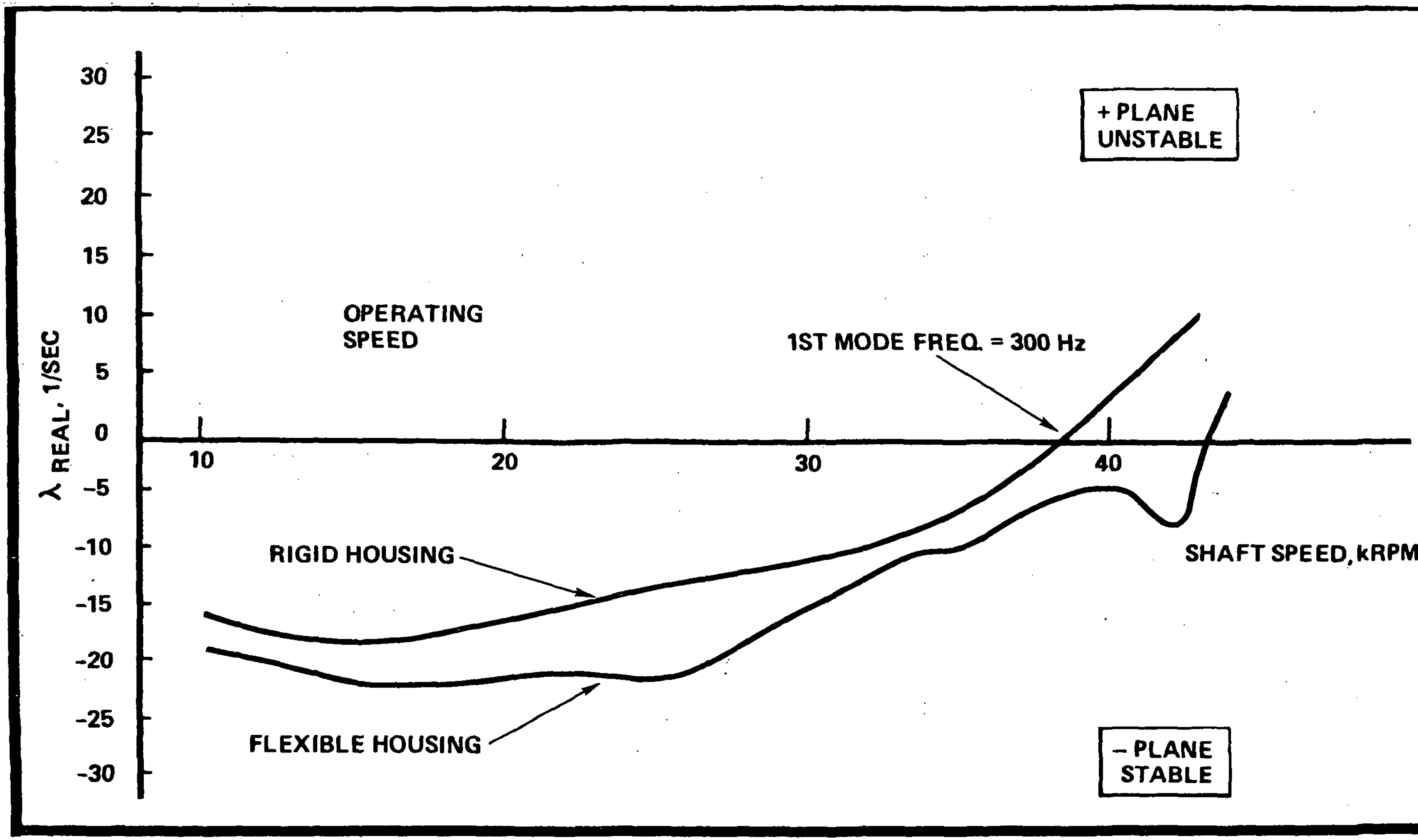


Figure 7. MK-380 (Redesign) Stability of First Rotor Mode

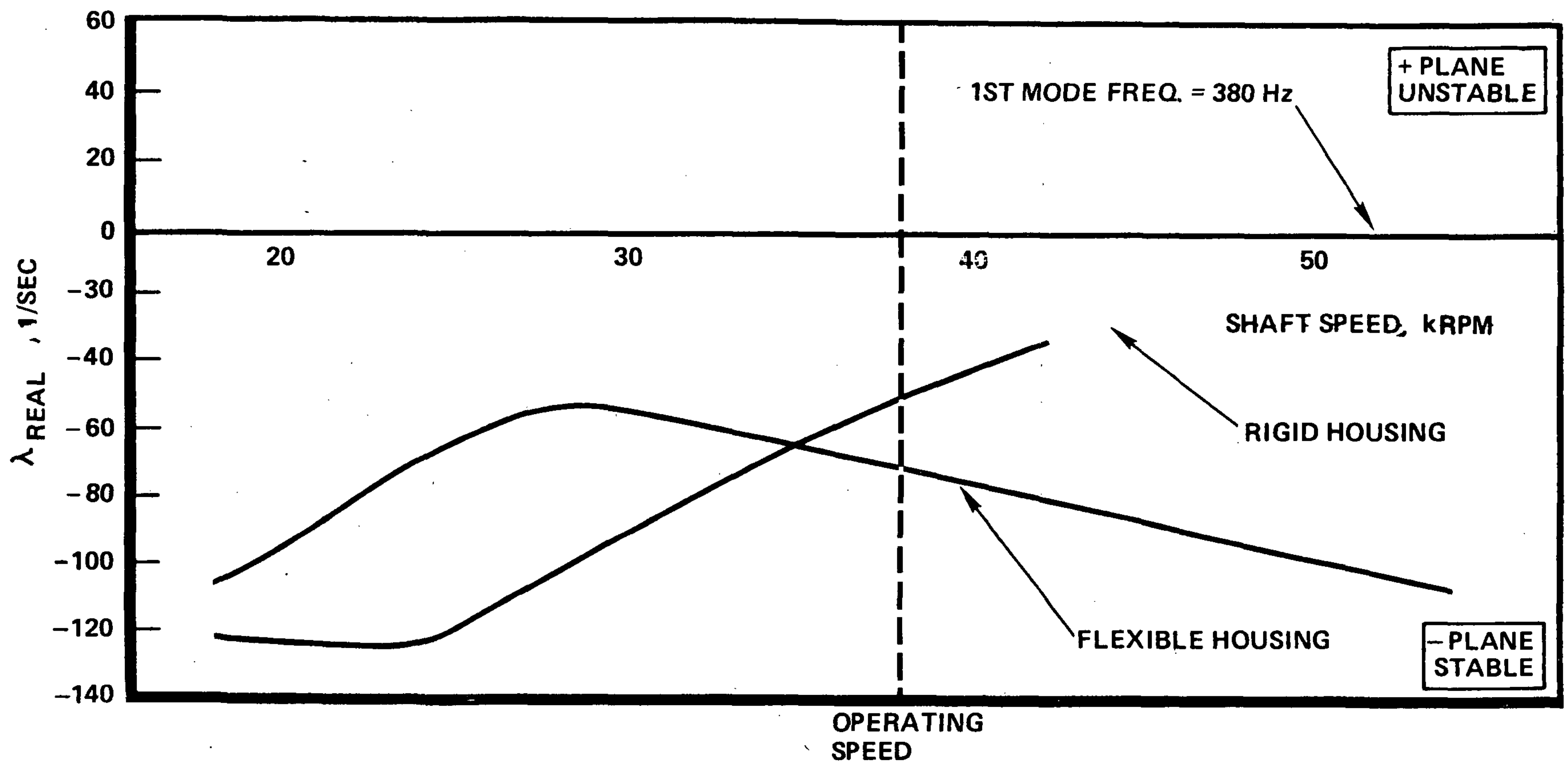


Figure 8. MK-38F Stability Analysis of First Rotor Mode Turbine Alford $\beta = 1.5$

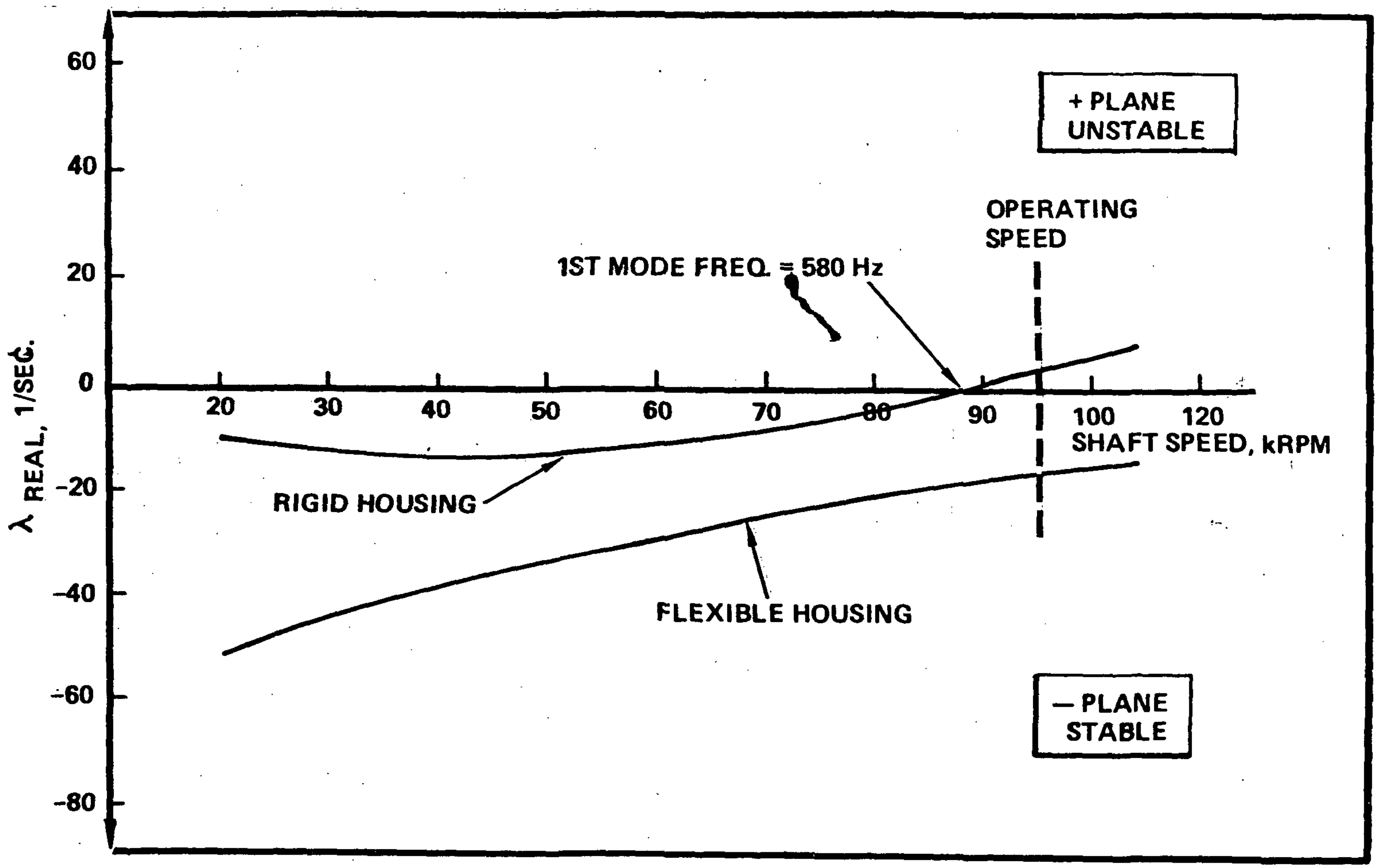


Figure 9. MK-49F Stability Map of First Rotor Mode Floating Ring Seal Inactive

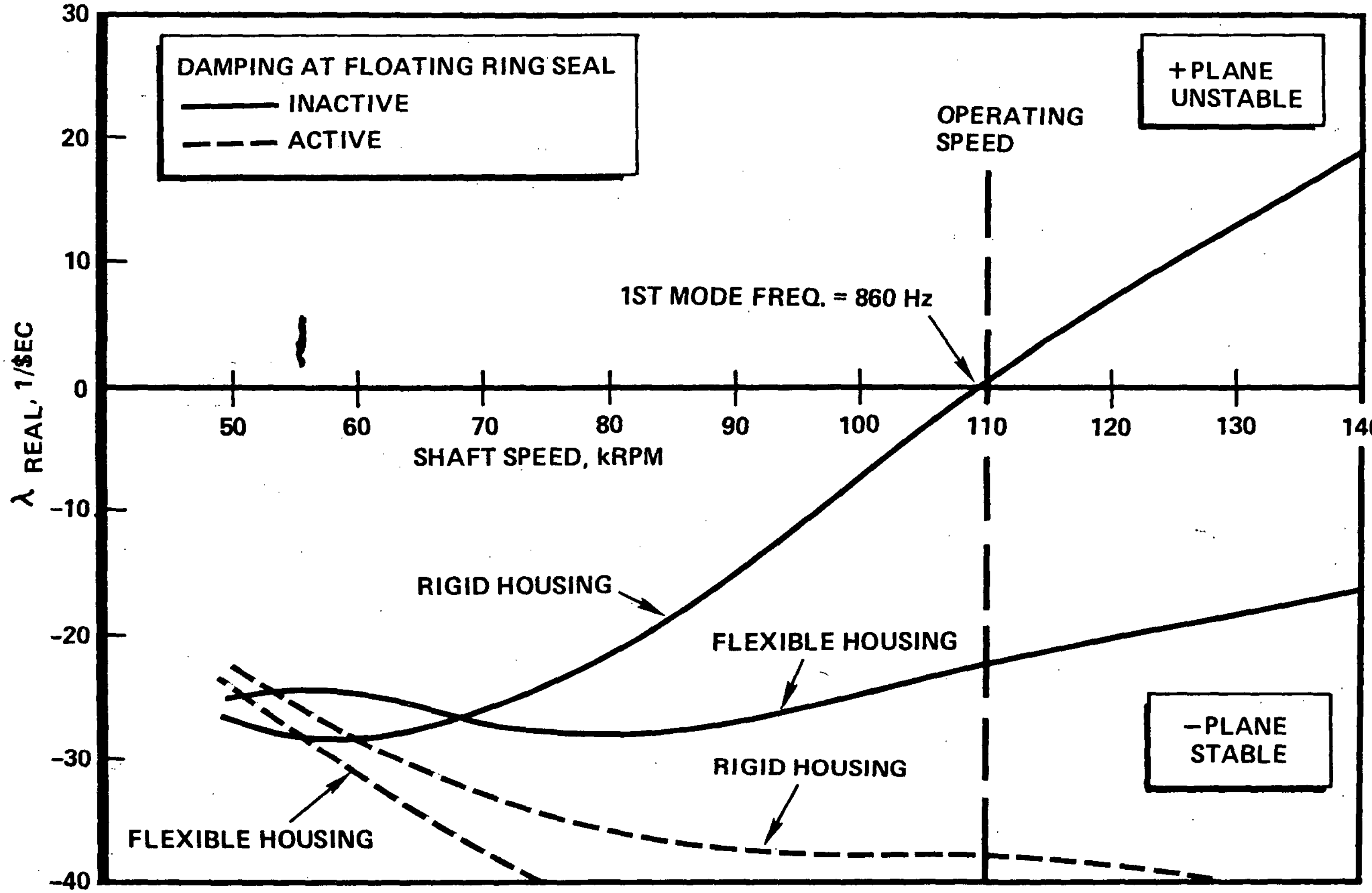


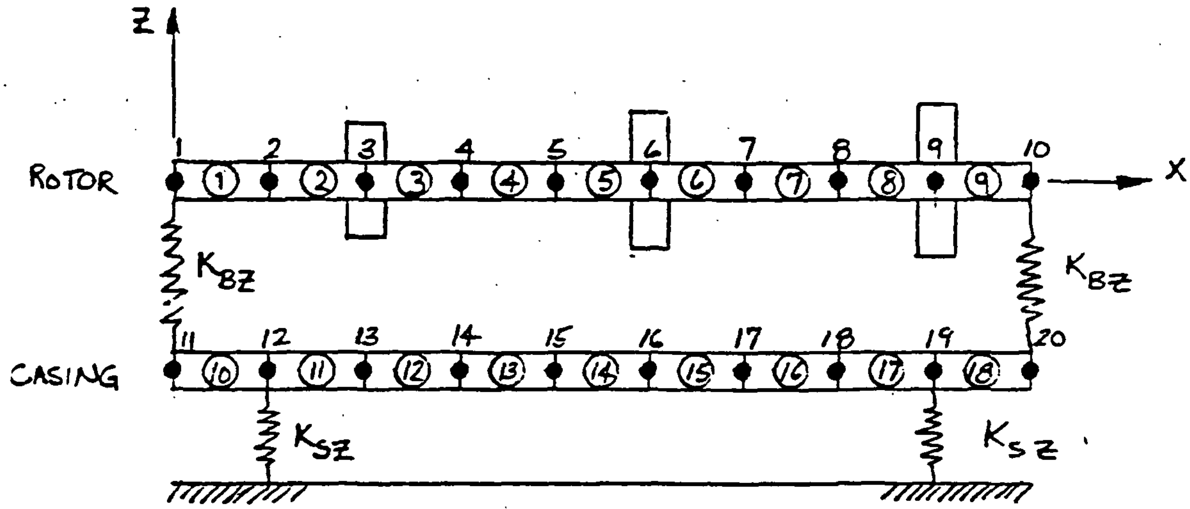
Figure 10. MK-49F Stability Map of First Rotor Mode Turbine Alford β = 0.25 Damping at Floating Ring Seal Inactive

In order to further evaluate how housing flexibility affects rotor stability, a simplified rotor-housing model was developed. This beam finite element model consists of 10-lumped-mass locations for both the rotor and housing. A description of the model is provided in Fig. 11. In order to account for the dynamic features significant to various turbopumps, three masses possibly simulating two impellers and a turbine disk were added. Generic mass and stiffness properties were selected and rotor/housing coupling coefficients from the HPOTP were chosen at random. These coefficients include the wear ring seal coupling on the first disk, the impeller cross-coupling coefficients on the second disk and the turbine interstage seal coefficients on the third disk. These typically representative coefficients are provided in Appendix A along with the resulting critical speeds, stability maps and the lowest rotor/housing mode shape.

A stability analysis of this simple model was performed with multiple variations of the housing. Critical speeds and stability information was calculated again using a rigid housing and various flexible housing conditions. It was determined that the asymmetry of the supports connecting the housing to ground directly affects the stability threshold. The stability threshold shows a dependence on the ratio of the asymmetry between the stiffness in the direction of the Y and Z axes. Furthermore, there is an optimum value of approximately 5:1 for the ratio of the asymmetric stiffnesses, as shown in Fig. 12. Each of these analyses was executed with a housing/rotor weight ratio of 6:1 which is generically representative.

Another study was conducted to examine the relationship between the housing/rotor weight ratios and the stability threshold. Using the simple model and varying only the weight of the housing, no relationship was observed between the weight ratios and the stability threshold as shown in Fig. 13.

A comparison of the results of the 20-lumped-mass model using a rigid housing versus a flexible housing, confirms that the rigid housing model is a conservative approach if the flexible housing meets certain prescribed conditions. Specifically, the support stiffness asymmetry ratio must be at least 3:1 and the housing/rotor weight ratio should be 6:1. The stability map of Fig. 14 compares the rigid housing model to the flexible housing model, with a 3:1 asymmetric support stiffness ratio.



SYMMETRIC IN Y AXIS

MODEL:

ROTOR

SHAFT DIAMETER: O.D. = 3.0", I.D. = 0.0"
MATERIAL: STEEL, E = 30 X 10^6 PSI
SIMULATED LUMPED MASSES: J13 = 10 LBS
J16 = 15 LBS
J19 = 20 LBS

JOINT LENGTH = 3.0"
ROTOR LENGTH = 27.0"

CASING

DIAMETER: O.D. = 15", I.D. = 14"
WEIGHT: 6/1 CASING/ROTOR WEIGHT RATIO

$K_{BZ} = K_{BY} = 500{,}000$ LB/IN

$K_{St} = 1.0 \times 10^7$, $1/3 = K_{SY}/K_{SZ}$

Figure 11. 20-Lump-Mass Turbopump Model

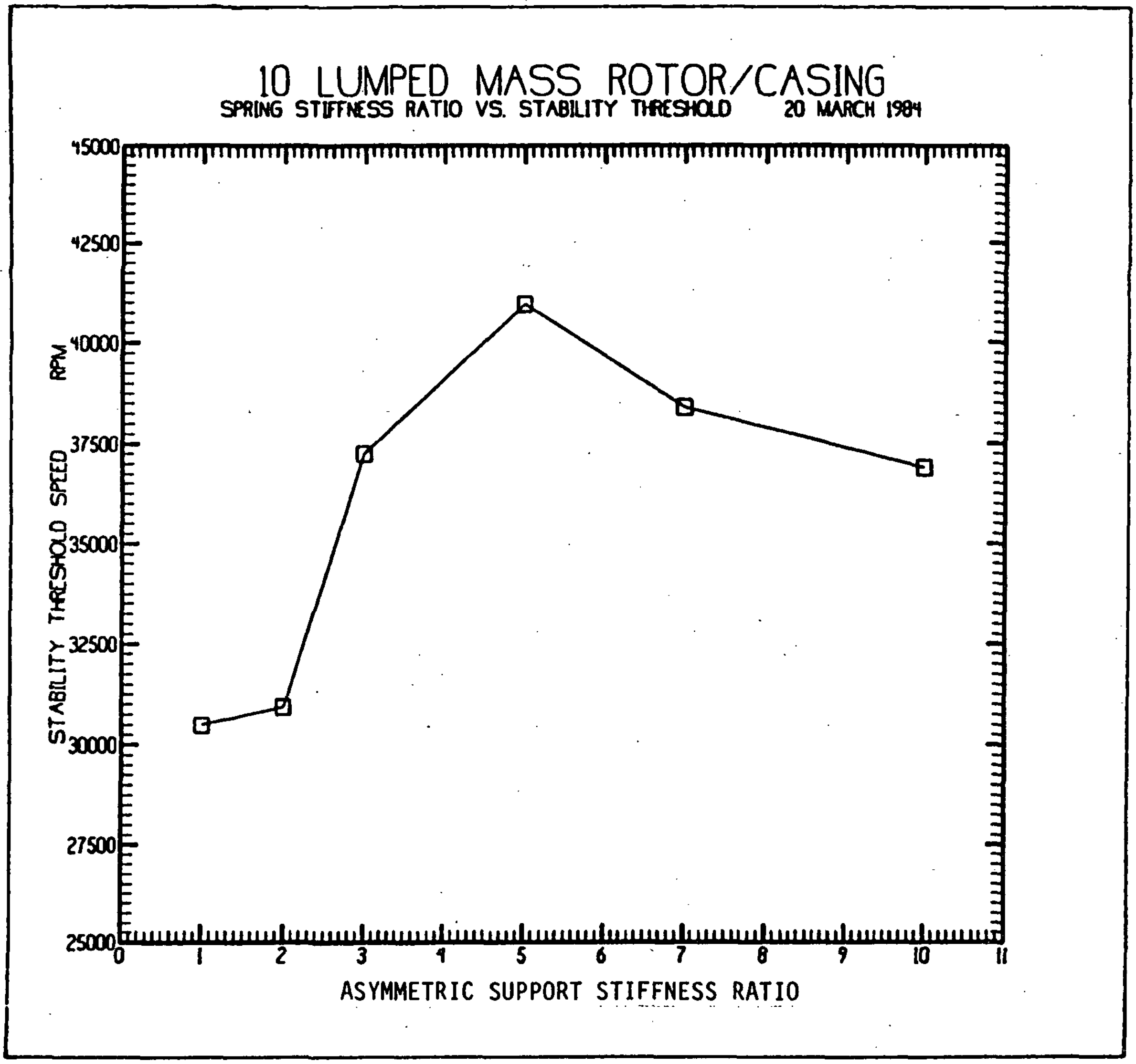


Figure 12. 10-Lumped-Mass Rotor/Casing Spring Stiffness Ratio vs Stability Threshold

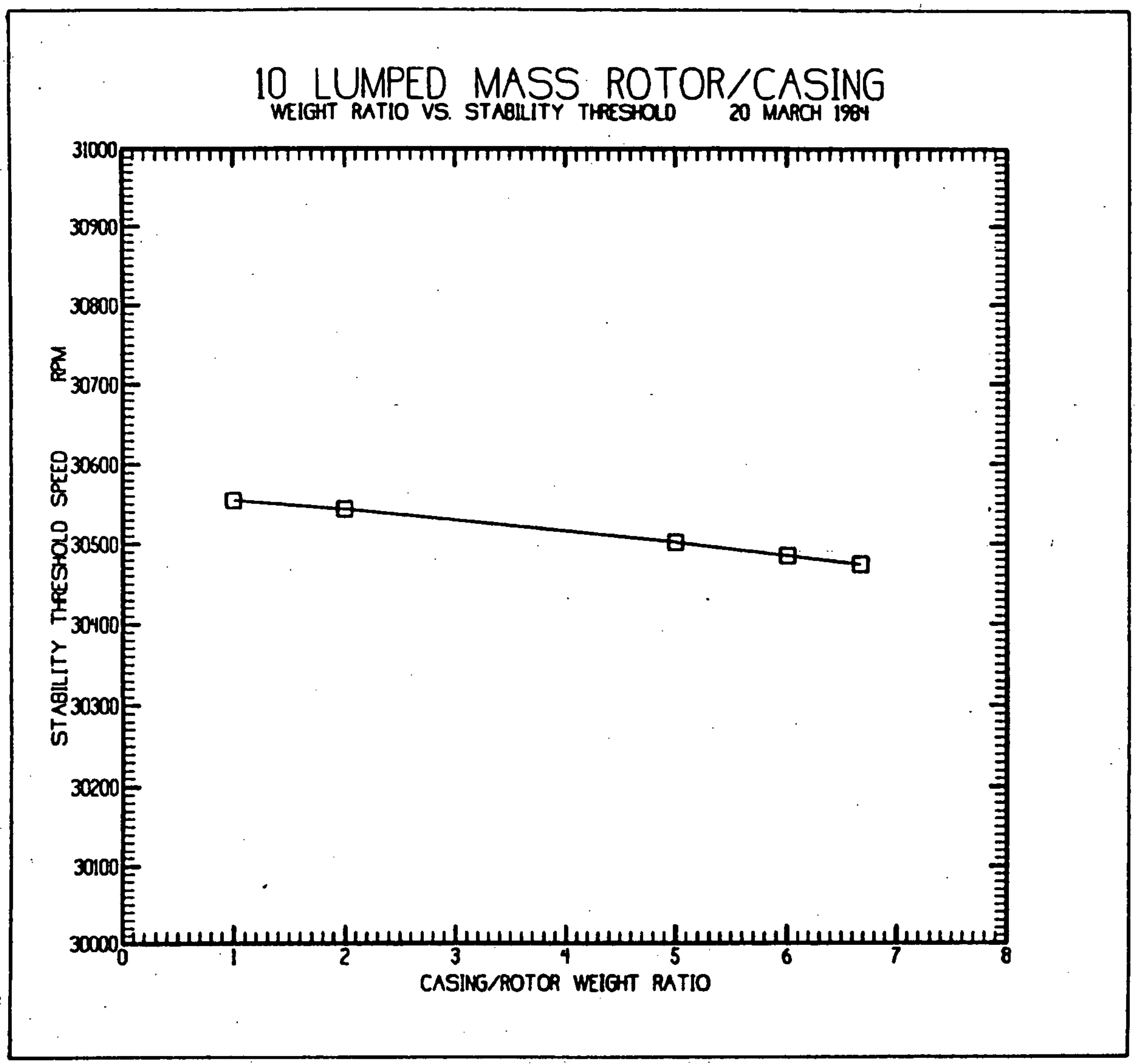


Figure 13. 10-Lumped-Mass Rotor/Casing Weight Ratio vs Stability Threshold

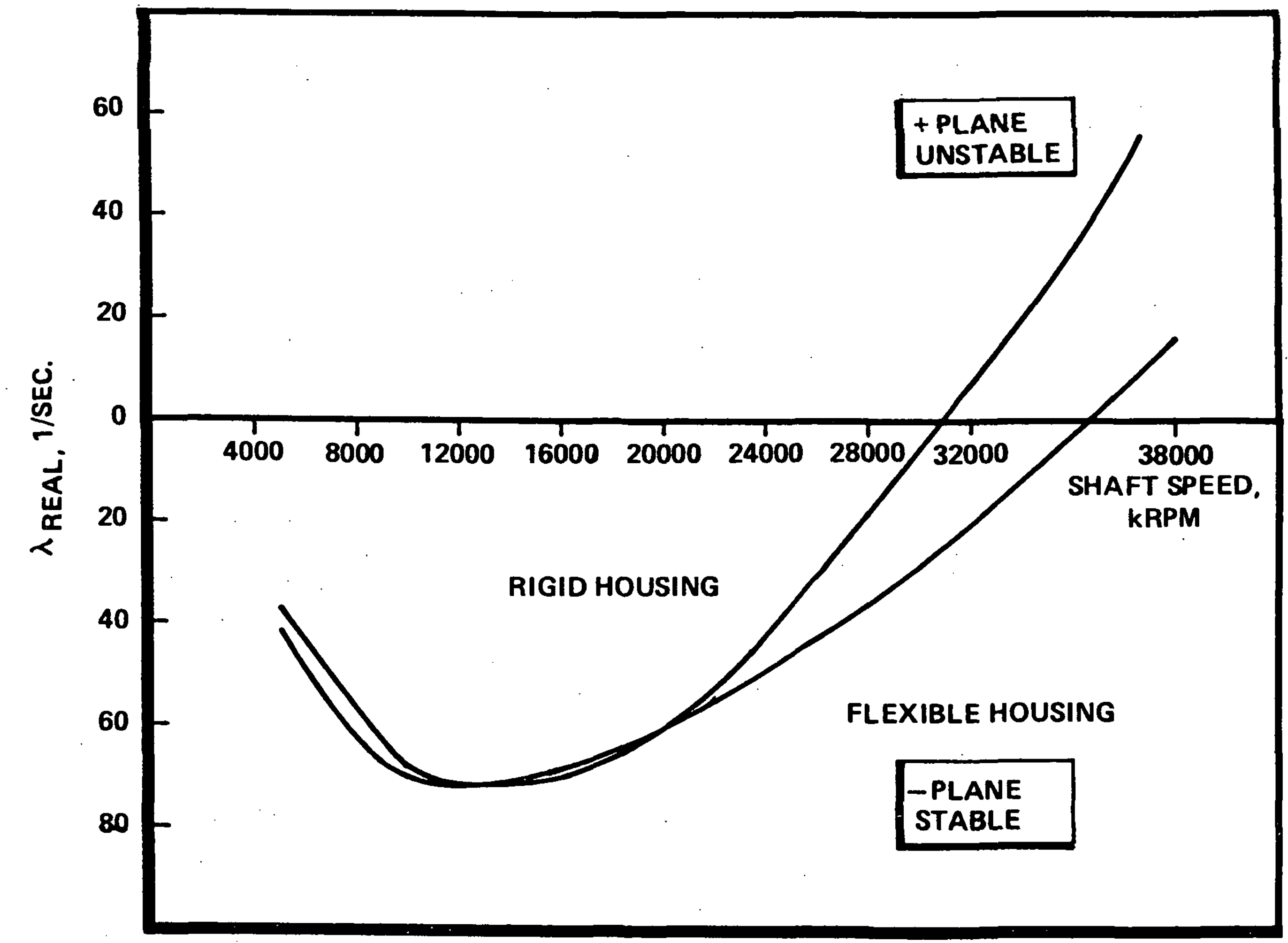


Figure 14. 10-Lumped-Mass Rotor/Casing Stability Map for First Rotor Mode

An analysis of six turbopumps demonstrates that asymmetric housing flexibility increases the rotor stability. A rigid housing assumption predicts an instability to occur prior to the same analysis with a flexible housing. Consequently, a rigid housing assumption is more conservative than a flexible housing model. The flexible housing acts as an energy absorber and dissipates some of the destabilizing forces in the rotor/housing system.

Also, a simple 20-lumped-mass turbopump model yields consistent results with the more complex, detailed turbopump models. That is, the rigid housing again renders conservative results. However, it is necessary to comply with certain prescribed conditions which are very typical of turbopumps. It was determined that the asymmetric spring stiffness ratio of the supports grounding the housing should be greater than or equal to 3. In addition, there is an optimum value of the asymmetric spring stiffness ratio of approximately 5. It is not fully understood at this time why the stability decreases with asymmetry greater than a 5:1 ratio. Another recommended condition is that the housing to rotor weight be at least 6:1 based on this weight ratio being representative of many existing turbopumps. Adherence to these conditions insures that the simple 20-lumped-mass turbopump model will yield consistent results with the more detailed models.

It follows that, in many cases, complex housing models need not be developed since a rigid housing gives comparable and more conservative results. As a result, time and money are conserved in the development of the rotordynamic model.

6.0 RSTAB ROTOR STABILITY ANALYSIS PROGRAM FOR A DESKTOP COMPUTER

A general rotor stability analysis program used to obtain modal stability (Root-Locus method) of turbomachinery with rpm-dependent rotor/casing coupling elements (e.g., close-clearance seals, Alford forces) was revised for application to this study. Figures 15 through 18 illustrate the principal procedures and subprograms of the existing code. Based on previous progress with regard to the effect of modal truncation on analysis accuracy in the operating speed range of interest, substantial reduction in program size requirements has been demonstrated to be possible while retaining the essential analytic capability to study the casing flexibility effects on rotor stability and bearing loads.

A general rewrite of routines was necessary, and segmentation of the analysis procedure into three well-defined steps (Fig. 19) greatly facilitated the adaptation of the resulting program to a desktop computer. The code has been fully tested and verified, and has demonstrated excellent agreement with equivalent mainframe computer programs.

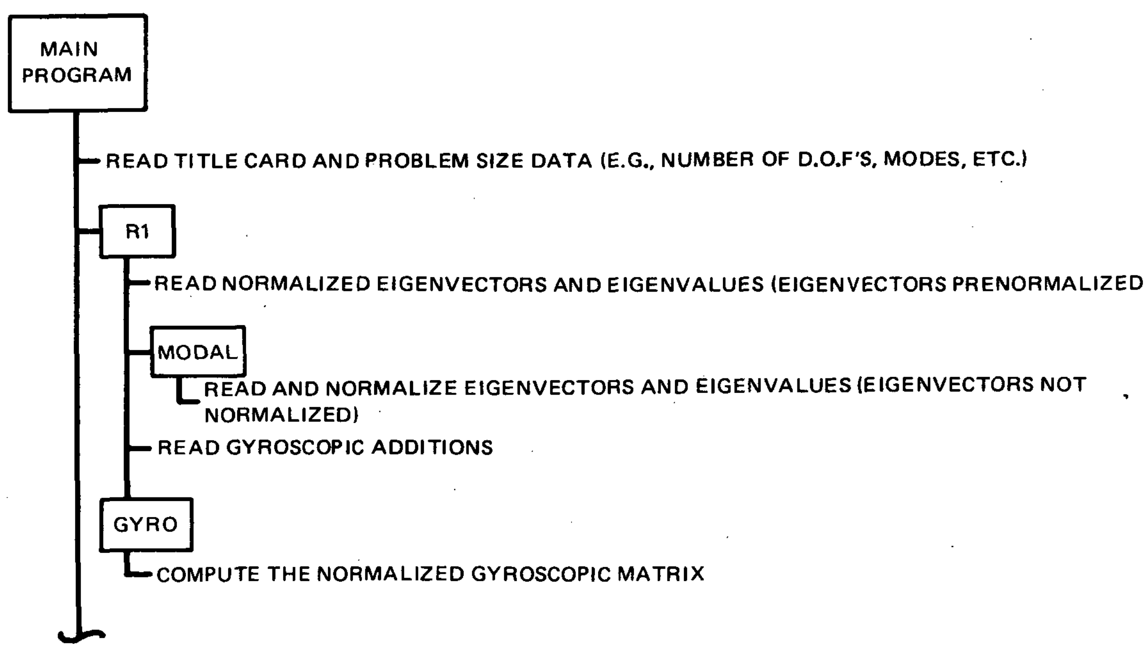


Figure 15. Rotor Stability Analysis Flowchart: Pre-Processing

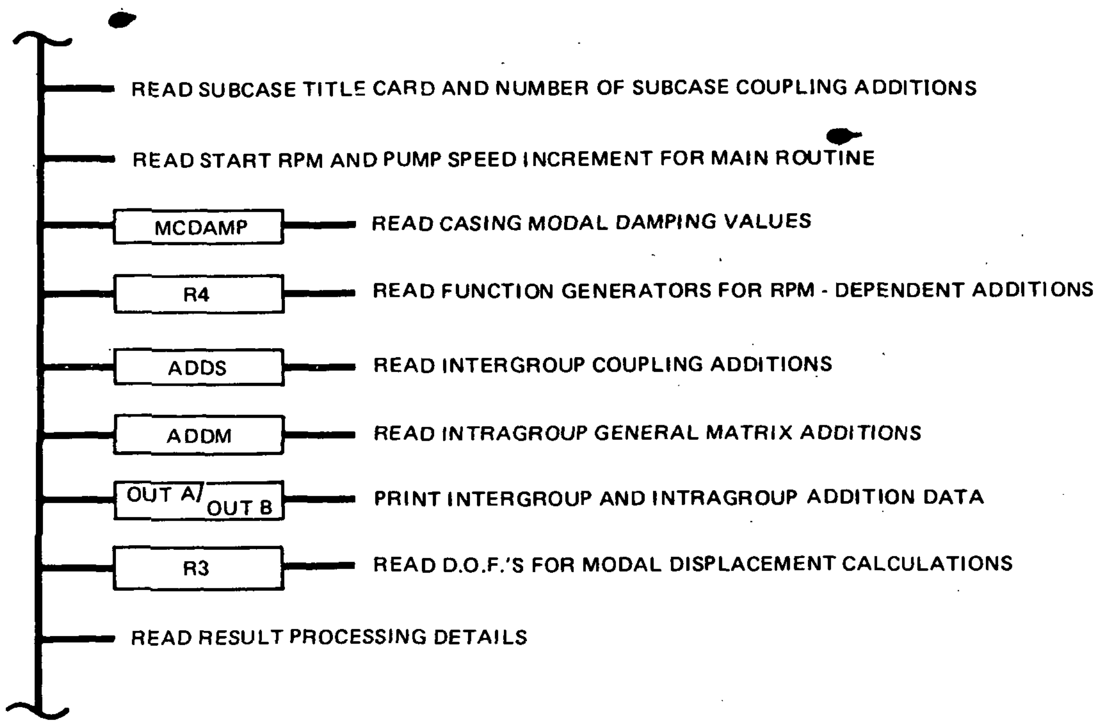


Figure 16. Rotor Stability Analysis Flowchart: Sub-Case Input

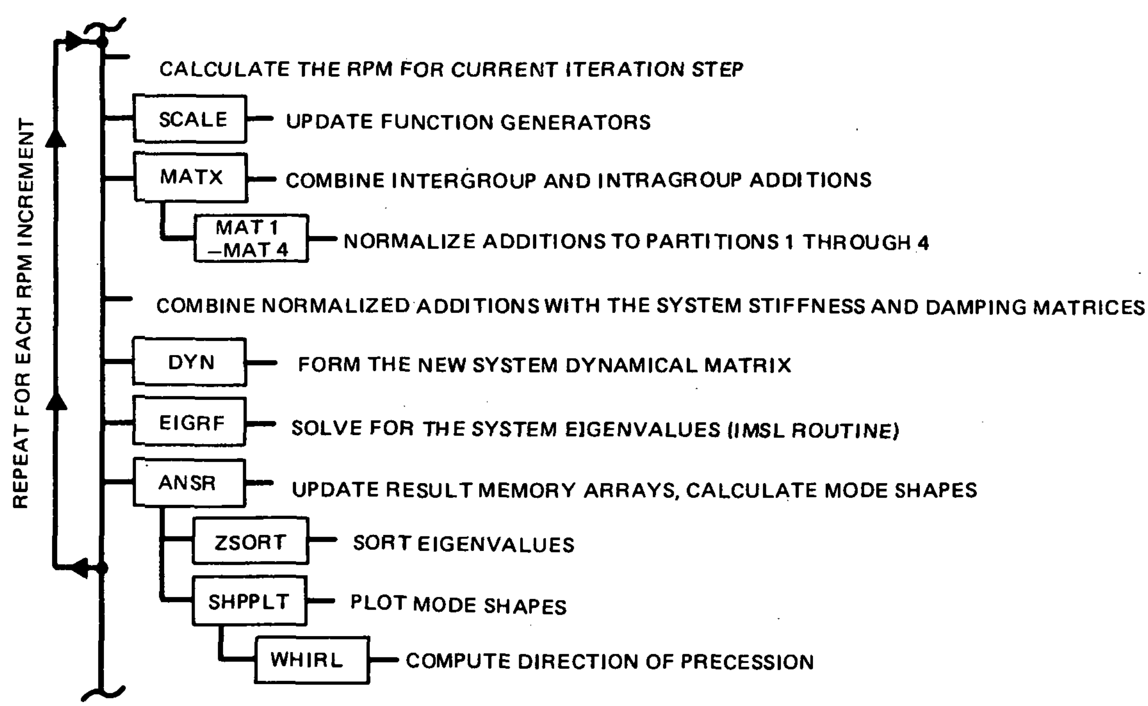


Figure 17. Rotor Stability Analysis Flowchart: Main Routine

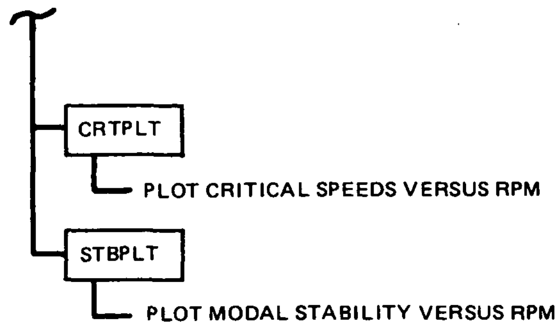


Figure 18. Rotor Stability Analysis Flowchart: Result Processing

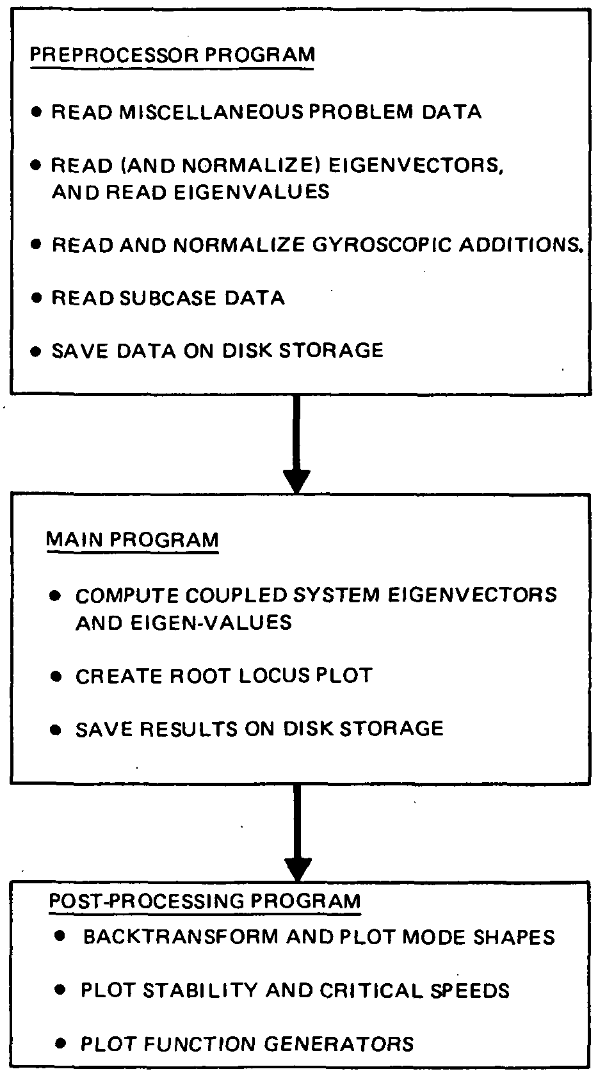


Figure 19. Rotor Stability Analysis Implementation on a Desktop Computer: Program Chaining to Improve Operational Efficiency

Some effort was directed toward the study of a variant matrix-method approach of determining the effect of casing flexibility changes on the rotor stability by considering their effect at a fixed rpm. This approach would take advantage of the matrix symmetry of casing flexibility changes and is used in parallel with the full-range analysis program.

Microsoft-FORTRAN Version 3.13 was used to develop the programs. The philosophy of program organization has been to enhance efficiency in the desktop computer environment and to provide clear subdivisions for chaining the programs to operate under main memory restrictions (the computer used for development was an IBM-PC with 256K bytes of main memory).

The code has been divided into three executable programs (Fig. 20 through 22), which are executed sequentially using the batch procedure RUNRSTAB. The first of these three (PRERSTAB) entirely preprocesses the input files, including all subcase iterations. It creates a binary run data file, which is then used by the main program (RSTAB). The main program, wich solves for the coupled system eigenvalues and mode shapes as operating speed is varied, produces a root-locus plot (upper half-plane) for each subcase run. This plot is the only graphics produced by the main program. All other results information is written to binary files for use by the postprocessor program (PSTRSTAB). This last program produces all remaining plotted output, including function generator curves, critical speed and stability plots and complex mode shapes. Restart capability is afforded by the various binary files created by each program, which can be saved for this purpose.

Details of critical speed and stability plot generation are provided in Appendix B. Finalized versions of these plots, and the function generator plots, are shown in Fig. 23 through 25. These plots were made on the IBM-PC using the PLOT 88 graphics library. As was previously described, the intersection of the modal natural frequency curve with the diagonal line representing synchronous speed (Fig. 23) indicates a potential critical speed. Also of interest is the intersection of the modal natural frequency with the half-synchronous speed line (also shown in Fig. 23). This indicates the potential subsynchronous response of the rotor first critical mode.

```
*PRERSTAB*************************************************************
C
C     P R O G R A M   P R E R S T A B
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C*********************************************************************
C
C     THE PRINCIPAL SUBROUTINES ARE USED IN THE FOLLOWING ORDER:
C
C     FORT01. . . . . . . . . . . . . READ PROBLEM SIZE DATA, READ AND
C                                     NORMALIZE GYROSCOPIC ADDITIONS
C            R1
C                  MODAL
C            TRANS
C
C     FORT02 . . . . . . . . . . . . . READ SUBCASE DATA, MATRIX DATA
C
C            ADDS
C            ADDM
C            OUTAB
C            R3
C
C*********************************************************************
C
C     INPUT FILE NAMES    DESCRIPTION
C     ----------------    ---------------------------------------------
C
C     <USER-DEFINEABLE>   FORMATTED USER'S INPUT FILE, INCLUDING ANY
C                         SUBCASE INPUT AND SUBSTRUCTURE MODE SHAPES
C
C     <USER-DEFINEABLE>   FORMATTED ROTOR NORMAL MODES (IF NOT INCLUDED
C                         IN THE USER'S INPUT FILE)
C
C     <USER-DEFINEABLE>   FORMATTED CASING NORMAL MODES (IF NOT INCLUDED
C                         IN THE USER'S INPUT FILE)
C
C     OUTPUT FILE NAMES   DESCRIPTION
C     -----------------   ---------------------------------------------
C
C     <USER-DEFINEABLE>   FORMATTED OUTPUT LISTING.  CAN BE 'PRN' (FOR
C                         IMMEDIATE PRINTING) OR ANY DISK FILE
C
C     RUNDATA.BIN         BINARY RUN DATA FILE FOR USE BY PROGRAM RSTAB
C
C     FGPLTS.BIN          BINARY FUNCTION GENERATOR DATA FOR PLOTTING BY
C                         PROGRAM PSTRSTAB
C
C     FIN.BAT             FORMATTED IBM BATCH FILE TO COPY OUTPUT
C                         LISTING FILES CREATED BY PROGRAMS RSTAB AND
C                         PSTRSTAB (IF NOT PRINTED IMMEDIATELY) TO THE
C                         USER-DEFINED OUTPUT LISTING FILE
C
C     LOG.SAV             BINARY USAGE LOG FOR PRODUCTIVITY ACCOUNTING
C
C*********************************************************************
```

Figure 20. Program PRERSTAB

```
*RSTAB*****************************************************************
C
C     P R O G R A M   R S T A B
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C**********************************************************************
C
C     THE PRINCIPAL SUBROUTINES ARE USED IN THE FOLLOWING ORDER:
C
C     PLOT . . . . . . . . . . . . . . SET UP ROOT LOCUS PLOT
C
C     FORT03 . . . . . . . . . . . . . SOLVE FOR EIGENVALUES VS. RPM
C
C             MATX
C             EIGRF
C             ANSR
C
C**********************************************************************
C
C     INPUT FILE NAMES    DESCRIPTION
C     ----------------    ---------------------------------------------
C
C     RUNDATA.BIN         BINARY RUN DATA FILE CREATED BY PROGRAM
C                         PRERSTAB
C
C     FIN.BAT             FORMATTED IBM BATCH FILE USED TO DETERMINE
C                         WHETHER TO PRINT THE OUTPUT LISTING DATA
C                         IMMEDIATELY, OR TO WRITE THE INFORMATION TO
C                         TEMPORARY FILE 'LIST2'
C
C     OUTPUT FILE NAMES   DESCRIPTION
C     -----------------   ---------------------------------------------
C
C     EIGENS.BIN          BINARY DATA FILE CONTAINING THE EIGENVALUES
C                         FOR EACH RPM STEP. USED BY PROGRAM PSTRSTAB
C                         FOR CRITICAL SPEED AND STABILITY PLOTTING
C
C     SHAPES.BIN          BINARY DATA FILE CONTAINING THE COMPLEX MODE
C                         SHAPES (IN NORMAL COORDINATES) FOR PLOTTING BY
C                         PROGRAM PSTRSTAB
C
C     LIST2               FORMATTED OUTPUT LISTING CREATED ONLY IF
C                         OUTPUT IS NOT PRINTED IMMEDIATELY
C
C*********************************************************************
```

Figure 21. Program RSTAB

```
*PSTRSTAB.FOR*************************************************************
C
C     P R O G R A M   P S T R S T A B
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C*************************************************************************
C
C     THE PRINCIPAL SUBROUTINES ARE USED IN THE FOLLOWING ORDER:
C
C     FNCPLT . . . . . . . . . . FUNCTION GENERATOR PLOTS
C
C     CRTPLT . . . . . . . . . . CRITICAL SPEED PLOTS
C
C     STBPLT . . . . . . . . . . STABILITY PLOTS
C
C     SHPPLT . . . . . . . . . . MODE SHAPE PLOTS
C
C*************************************************************************
C
C     INPUT FILE NAMES    DESCRIPTION
C     ----------------    ---------------------------------------------
C
C     FGPLTS.BIN          BINARY FILE CREATED BY PROGRAM PRERSTAB
C                         CONTAINING THE FUCTION GENERATOR DATA FOR
C                         PLOTTING
C
C     EIGENS.BIN          BINARY FILE CREATED BY PROGRAM RSTAB
C                         CONTAINING THE EIGENVALUES AT EACH RPM STEP
C                         FOR CRITICAL SPEED AND STABILITY PLOTTING
C
C     SHAPES.BIN          BINARY FILE CREATED BY PROGRAM RSTAB
C                         CONTAINING THE COMPLEX MODE SHAPES FOR MODE
C                         SHAPE PLOTTING
C
C     FIN.BAT             FORMATTED IBM BATCH FILE USED TO DETERMINE
C                         WHETHER TO PRINT THE OUTPUT LISTING DATA
C                         IMMEDIATELY, OR TO WRITE THE INFORMATION TO
C                         TEMPORARY FILE 'LIST3'
C
C     OUTPUT FILE NAMES   DESCRIPTION
C     -----------------   ---------------------------------------------
C
C     LIST3               FORMATTED OUTPUT LISTING CREATED ONLY IF
C                         OUTPUT IS NOT PRINTED IMMEDIATELY
C
C     VECT.TMP,MAPS.TMP   BINARY TEMPORARY FILES CREATED BY THE PLOTT88
C                         LIBRARY. THESE WILL BE DELETED UPON NORMAL
C                         TERMINATION OF THE PLOTT88 ROUTINES
C
C*************************************************************************
```

Figure 22. Program PSTRSTAB

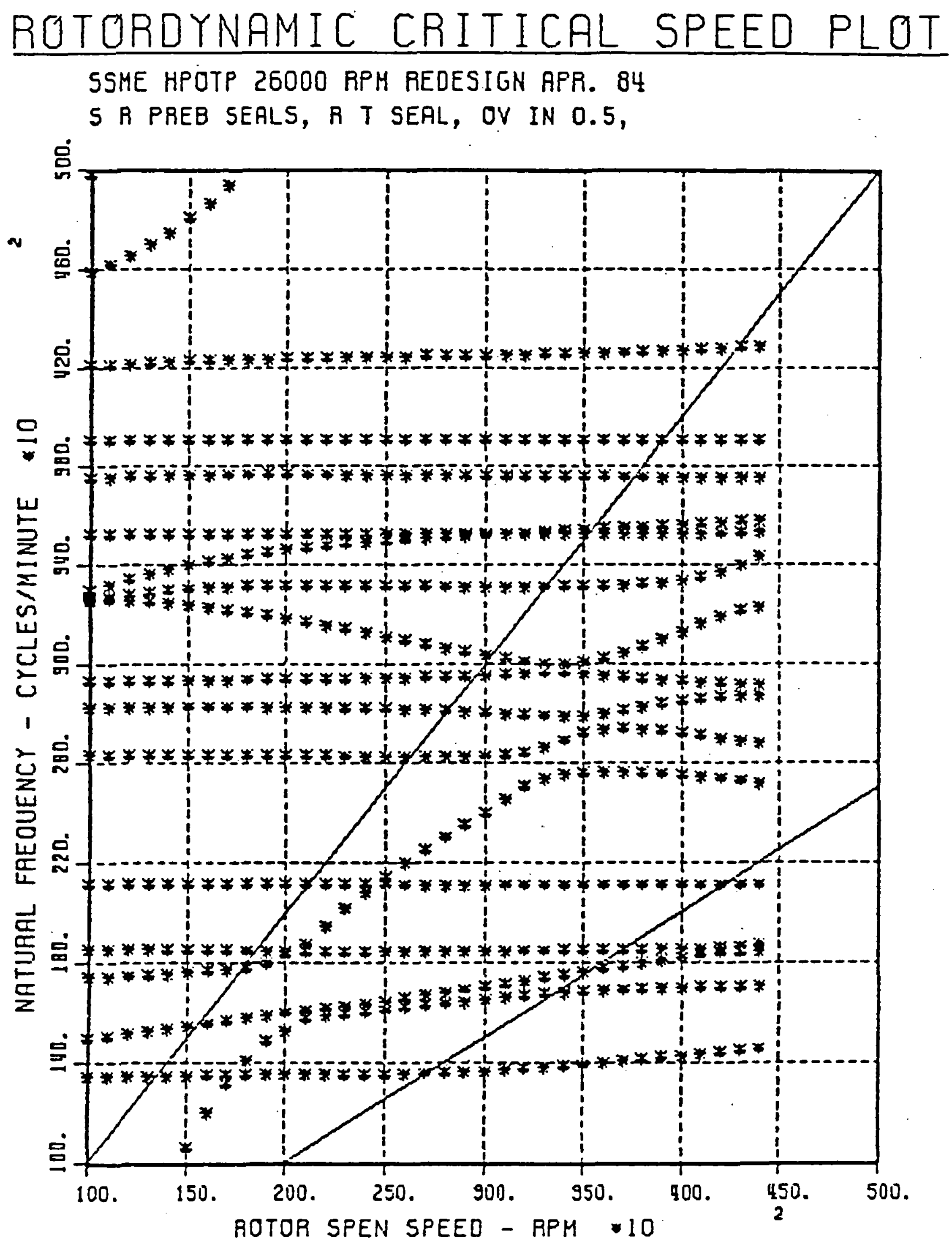


Figure 23. Rotor Stability Analysis Program Critical Speed Plot

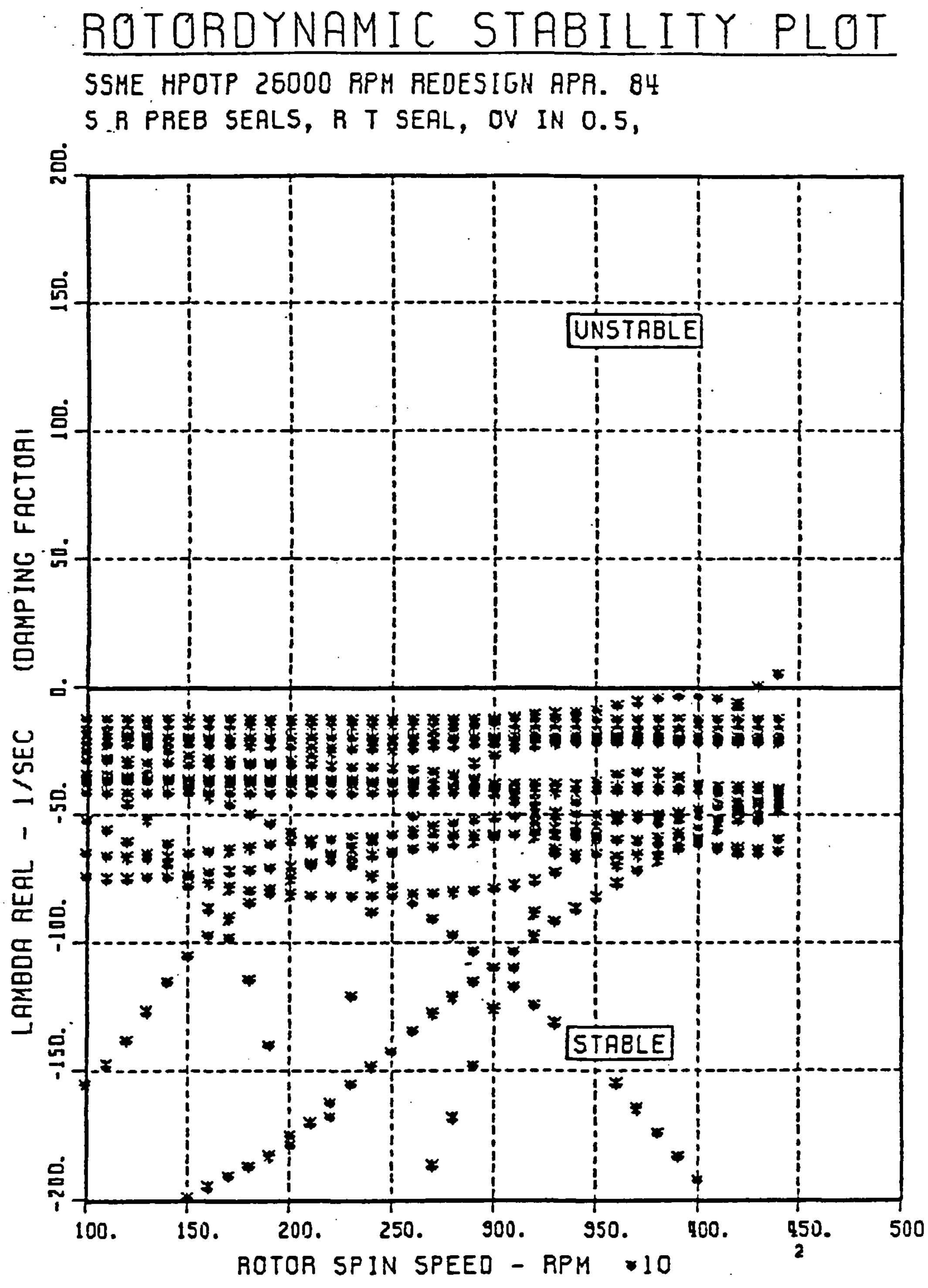


Figure 24. Rotor Stability Analysis Program Stability Plot

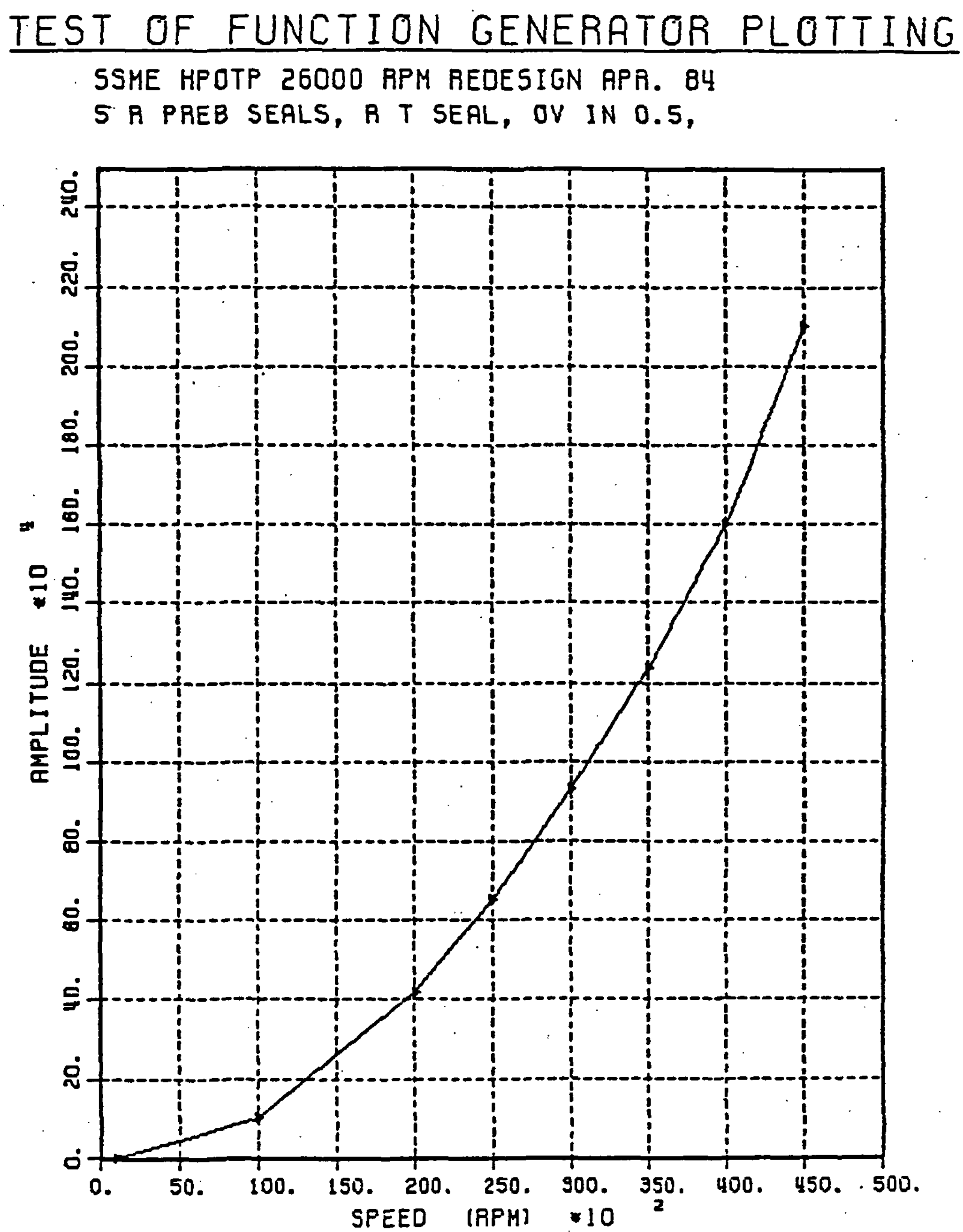


Figure 25. Rotor Stability Analysis Program Function Generator Plot

A root-locus plot routine for the main program unit was developed. The plot is generated while the program executes and the complex eigenvalues are computed. The root-locus plot is a useful tool in examining turbopump dynamics. This capability also allows good visual feedback on execution status which is particularly valuable in the interactive mode of operation.

An example of a root-locus plot generated by RSTAB is provided as Fig. 26. The crossover of the rotor first critical mode into the positive real half-plane at half-synchronous speed is a clear indication of whirl instability initiation.

The critical damping ratios of the modes can also be graphically determined with the root-locus plot. The complex eigenvalue is related to the critical damping ratio by the following equation:

$$\mathrm{Re}[\lambda_i] = \zeta_i \sqrt{\mathrm{Re}[\lambda_i]^2 + \mathrm{Im}[\lambda_i]^2}$$

where:

λ_i = complex eigenvalue of the i<u>th</u> mode
ζ_i = critical damping ratio of the i<u>th</u> mode
$\mathrm{Re}[\lambda_i]$, $\mathrm{Im}[\lambda_i]$ = real, imaginary components of λ_i

On a root-locus plot, lines of equal damping are straight lines extending radially from the origin. Lines of equal damping for 1, 2.5, 5, and 10% of critical damping are shown on the plot for indication purposes (exact calculation of the damping ratio for any mode is easily made by referring to the data listed on the output file by RSTAB for each rpm increment).

Appendix B contains background on modal modeling while Appendix C includes a data/output listing of typical model results. A users manual is contained as Appendix D.

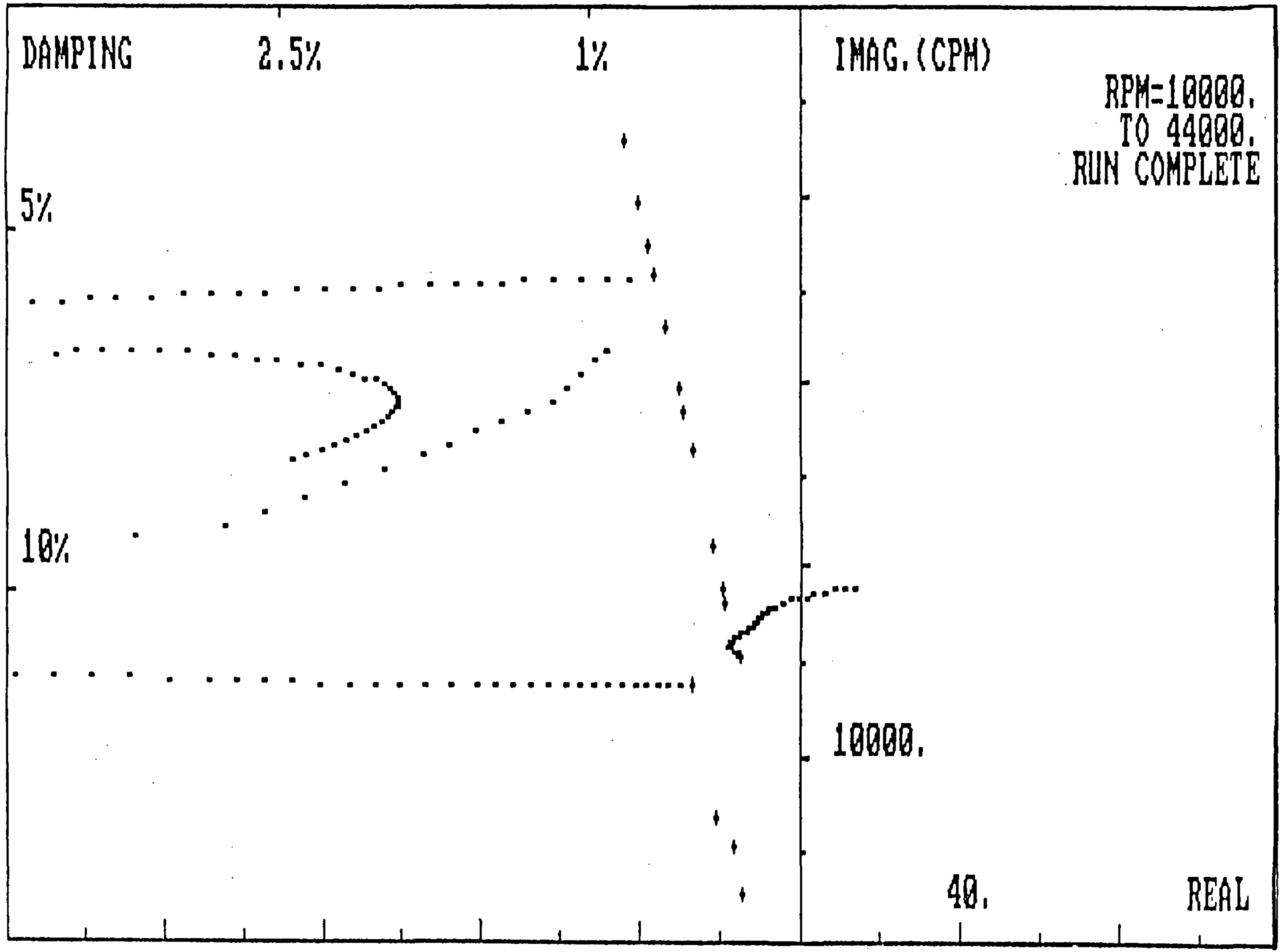


Figure 26. Rotor Stability Analysis Program Root-Locus Plot

7.0 USE OF THE ROOT LOCUS IN STABILITY STUDIES

Of the several basic methods used in control system analysis, the one that appears to be most adaptable is the Root Locus. Basically, it examines changes in the roots of the characteristic equation as changes are made in some parameter. The technique is commonly used in linear systems where the components are described in terms of the Laplace operator, S. The characteristic equation is a polynomial in S and roots of a polynomial can be written as:

$$S = R + Ij$$

The real portion of the root (R) indicates exponential activity, i.e., exp (Rt). Negative values of R indicate decay with time, while positive values indicate divergence with time. The imaginary portion of the root (I) indicate oscillatory activity. A root with non-zero values for R and I indicate oscillatory response with an exponential envelope.

The concept of the root locus is to present the characteristic equation in such a form that for changes in a specific parameter, the variation in the resulting roots can be easily seen. A graphical approach is used so that resonant frequencies and tendencies toward instability due to parameter changes are easily seen.

7.1 AN EXAMPLE OF THE APPROACH

As an example, consider a characteristic equation of the form:

$$A + BS + CS^2 + DS^3 + S^4 = 0$$

where

$$A = A_1 K$$
$$B = B_o + B_1 K$$
$$C = C_o + C_1 K$$

The equation can be separated into two polynomials, one of which has K as a factor. It can then be expressed as a ratio of polynomials which are factored:

$$[A_1 + B_1 S + C_1 S^2] K + S[B_o + C_o S + DS^2 + S^3] = 0$$

or

$$\frac{KC_1[S^2 + B_1 S/C_1 + A_1/C_1]}{S[S^3 + DS^2 + C_o S + B_o]} = -1$$

or

$$\frac{KC_1[S + R_1 + I_1 j][S + R_1 - I_1 j]}{S[S + R_2][S + R_3 + I_3 j][S + R_3 - I_3 j]} = 1 \angle -180 \text{ degrees}$$

For any value of S(= r + ij), each factor of the numerator and denominator can be expressed as a magnitude and angle so the left side can be expressed as a single magnitude and angle. For a value of S to be a root (for any value of K greater than 0 and less than infinity) the angle must be -180 degrees. The value of K can then be chosen to produce unity magnitude. A numerical example is shown in Fig. 27.

The position of roots of the numerator (zeros) are shown as "0" while the roots of the denominator (poles) are shown as "X." When K = 0, the roots of the characteristic equation are the poles. As K is increased, the roots migrate along loci as shown. The figure shows that for this system, the damping for the resonant roots is decreased and actually becomes unstable (positive real value for root) as K is increased. In addition, the increasing value of K results in an additional resonance with significant damping. The figure shows how the angle summation is obtained for a typical point shown as "□."

While this technique can be used to obtain quantitative solutions, its great strength is to quickly assess concepts of coupling and the effect of small changes in components. It is obvious, for example, that if the resonant system poles were initially much closer to the imaginary axis, much smaller values of K would be allowed if stability is to be maintained.

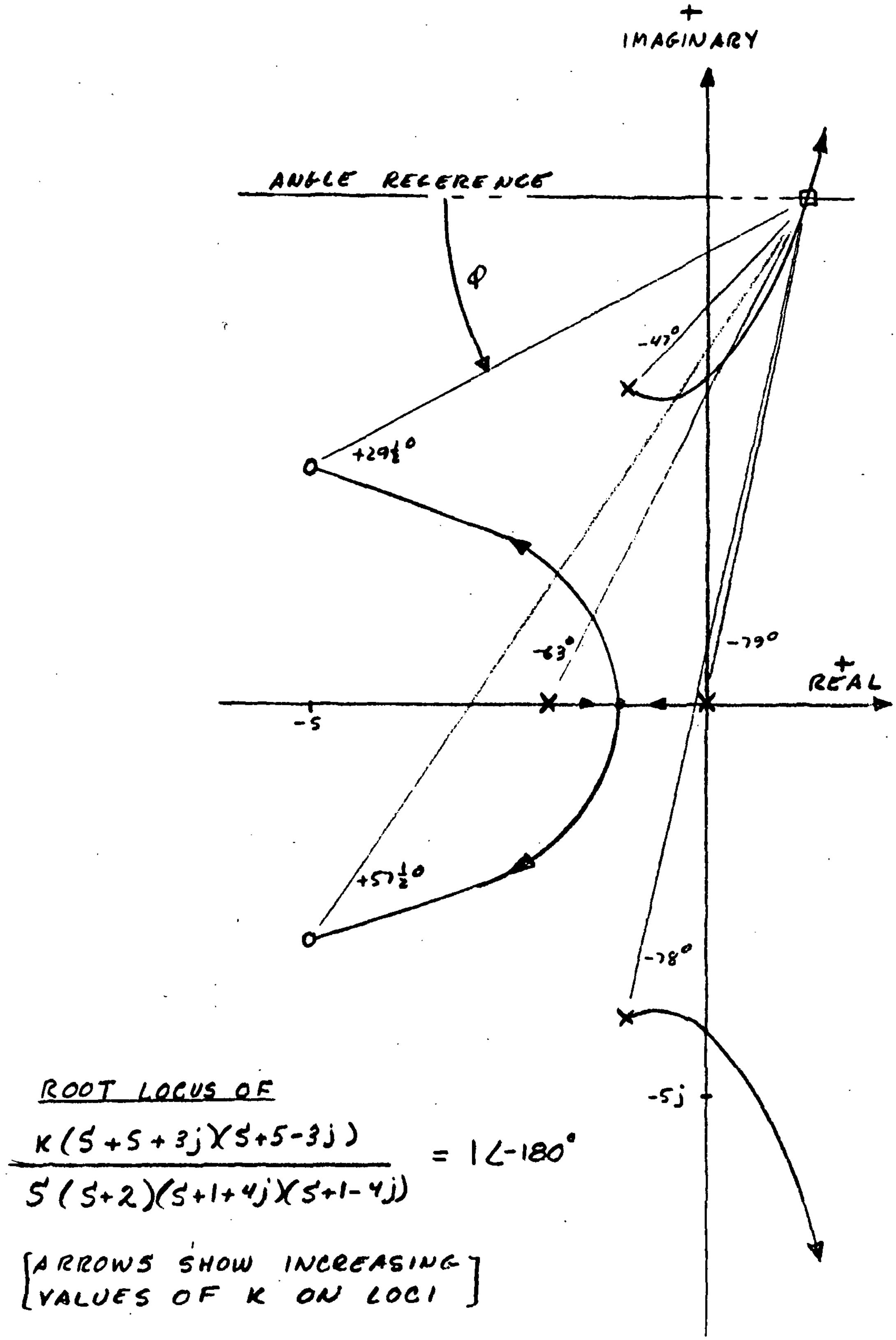


Figure 27. A Typical Root Locus Diagram

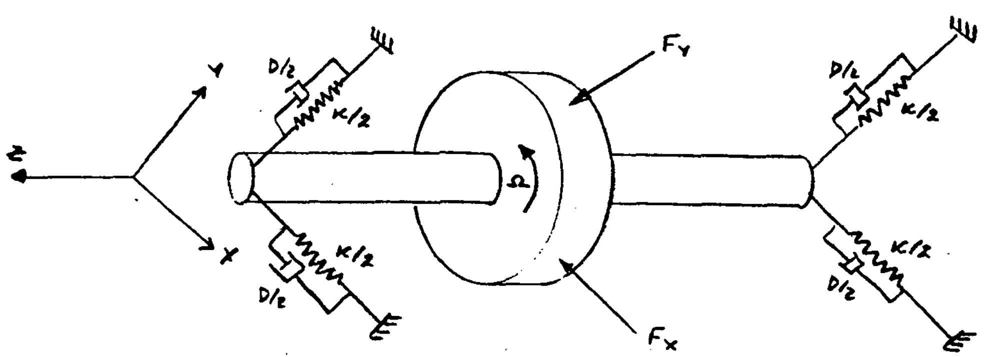
7.2 APPLICATION OF ROOT LOCUS METHODS TO A ROTOR

While the normal Root Locus techniques can be applied directly to many structural and control system problems, a basic difference in the system equations exists in rotating machinery. This difference is in cross coupling between axes. It is this coupling that can lead to rotor instability. This type of coupling is most often found in gyroscopics, fluid bearings and seals, and in turbines.

As a first step to evaluating the utility of the Root Locus we use a Jeffcott rotor as shown in Fig. 28. The equivalent bearing-shaft dynamics in the X-Z and Y-Z are shown as springs and dashpots to ground at both ends of the symmetrical rotor. The restoring forces directly applied to the central mass are typical of equations for a shaft seal.

For this axisymmetrical case a coordinate transformation may be made.

$$Y = -jx$$
$$X = jY$$



EQUATIONS OF MOTION IN Z-X AND Z-Y PLANES

$$(Ms^2 + Ds + K)X = F_X = -(K' + D's)X - \frac{\Omega}{2}D'Y$$
$$(Ms^2 + Ds + K)Y = F_Y = -(K' + D's)Y + \frac{\Omega}{2}D'X$$

WHERE $s \approx d/dt$

Figure 28. Jeffcott Rotor With Seal Cross Coupling

The characteristic equation of motion for either the X-Z or Y-Z plane can then be written as either:

$$[MS^2 + (D+D')S + (K+K')] = \frac{\Omega}{2} D'j$$

or

$$\frac{\Omega D'/2M}{[S^2 + 2\xi\omega S + \omega^2]} = -j = 1 \angle -90 \text{ degrees}$$

The second form of the characteristic equation is similar to the standard Root Locus form except that for S to be a root, an angle criterion of -90 degrees rather than -180 degrees must be satisfied.

In Fig. 29, the root loci for this system are shown. The in-place resonance (complex poles) are assumed to have a resonant frequency of 3000 rad/sec and 10% of critical damping at zero speed ($\Omega = 0$). As speed is increased, it can be seen that use of 90 degree criterion results in roots which are no longer complex pairs. The root in the upper half plane (corresponding to forward precession) shows decreased stability with increased speed while the lower root (backward precession) shows increased stability. The speed at which neutral stability occurs is obtained from:

$$\Omega D'/2M = 300 \times 6000$$

It may be interesting to note that more standard rotor dynamics approaches solve both of the equations shown in Fig. 28 simultaneously. If we, in fact, do this for the simple case we obtain a characteristic equation of the form:

$$[MS^2 + (D+D')S + (K+K')]^2 = -\left(\frac{\Omega D'}{2}\right)^2$$

or

$$\frac{(\Omega D'/2M)^2}{[S^2 + 2\xi\omega S + \omega^2]^2} = -1$$

A Root Locus of the equation is very similar to Fig. 29, except that it shows twice the number of loci. Half the roots are realistic, while half are nonrealistic reflections of the true roots about the real axis. There is no direct indication as to which roots are realistic. The problem here is that the in-plane dynamics are squared and speed is squared. All information as to the direction of rotation is lost. The roots of the equation include all possible roots for both positive and negative rotation. For a real pump, rotation is only in one direction and only half the roots will be of interest.

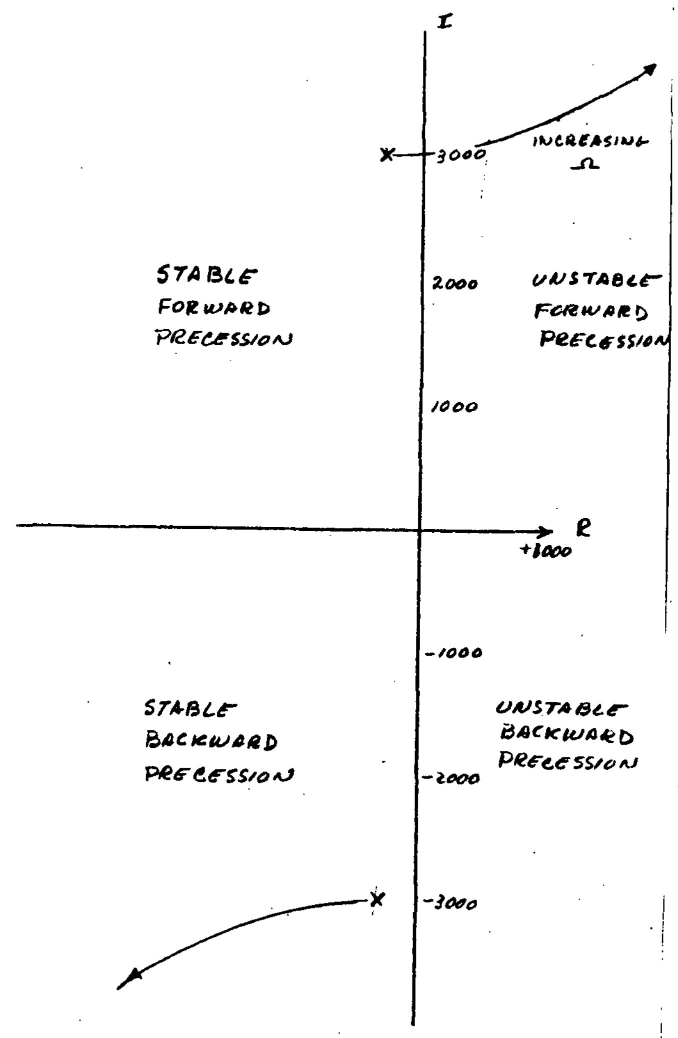


Figure 29. Root Locus of a Jeffcott Rotor With Seal Cross Coupling

An example of how this approach can be used in a realistic situation is indicated by the configuration shown in Fig. 30. Here a symmetrical rotor is mounted on a rigid casing which is supported to ground with springs and dashpots. In this case, a two step approach is used. Step 1 essentially determines the roots of the in-plane dynamics with no cross coupling. The second step determines the destabilizing effect of the cross coupling. Although Root Locus approaches were used for both steps (Fig. 31), the more common eigenvalue approach would be of advantage for the in-plane dynamics and the -90 degree Root Locus would be used for the last step to demonstrate the speed-stability aspects.

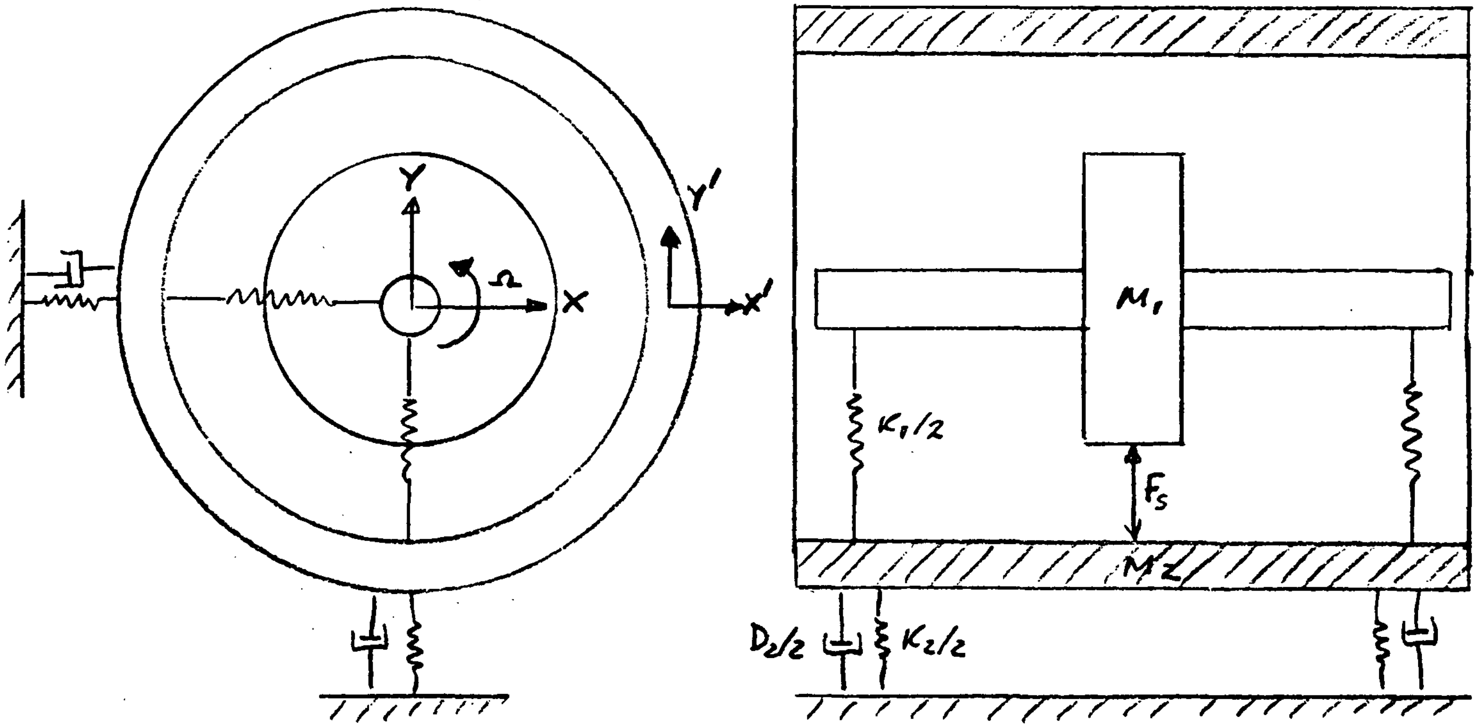
7.3 A GENERIC HIGH-SPEED ROTOR-CASING MODEL

The rotor model shown in Fig. 32 was used as a basis for testing the various analytical approaches. Basically, it is an overhung turbine configuration similar to the SSME high pressure liquid oxygen pump. A pump with both turbine and impeller located inboard of the bearings can be simulated by combining their inertia and cross coupling coefficients as the model impeller and deleting the model turbine mass properties.

As shown in Fig. 32, there are eight masses, each with inertia in translation and angulation. The casing is modeled as a uniform beam of total length, L, and section modulus, I, in bending. The casing was divided into three beams to obtain stiffnesses. Masses and inertias for the end masses are half the values used for the inboard masses.

The rotor shaft was divided in the same way as the casing. The impeller (mass 11, 12) and the turbine (mass 15, 16) are assumed to be disks added to the shaft and described by their weight and radius of gyration. Gyroscopic and seal faces are assumed to act only at the stations of the impeller and turbine.

Although the model is somewhat restrictive, it has enough generality to demonstrate the significant interactions one would expect in a rotor design. The restrictions simplify solutions and minimize computer time and allow more productive interactive computer sessions.



SEAL $\quad F_x = -(K' + D's)(x - x') - \frac{\Omega}{2} D'(Y - Y')$

SYSTEM EQUATION OF MOTION

$$\underbrace{\underbrace{M_1 s^2 [M_2 s^2 + (D_2 + D')s + (K_1 + K_2 + K')] + [K_1 + K' + D's][M_2 s^2 + D_2 s + K_2]}_{\text{STEP \#1 OBTAIN ROOTS WITH } -180^\circ \text{ ROOT LOCUS}} = \frac{\Omega}{2} D' M_2 s^2 j}_{\text{STEP \#2 OBTAIN ROOTS WITH } -90^\circ \text{ ROOT LOCUS}}$$

Figure 30. Jeffcott Rotor With Seal Cross Coupling and Casing Motion

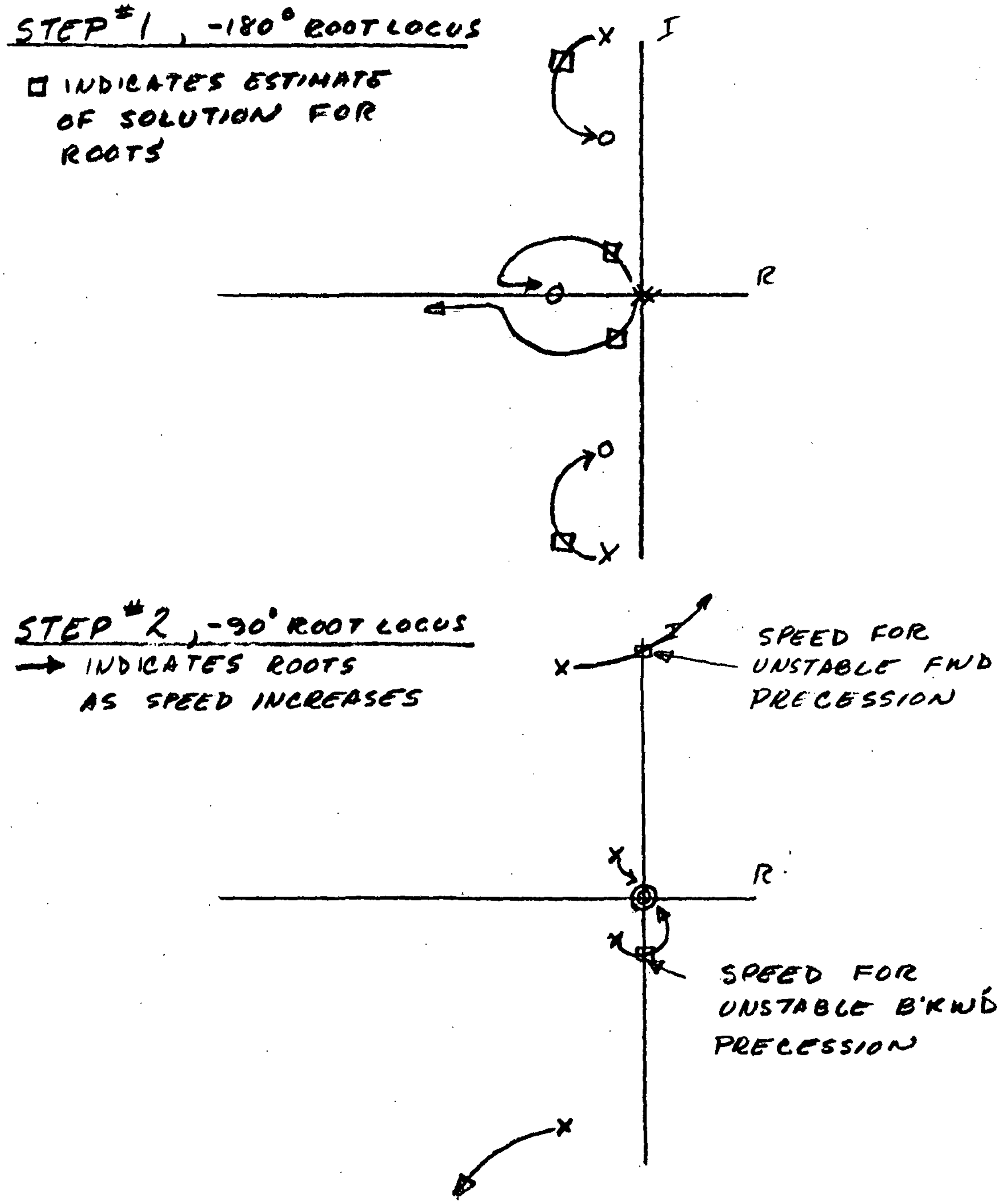
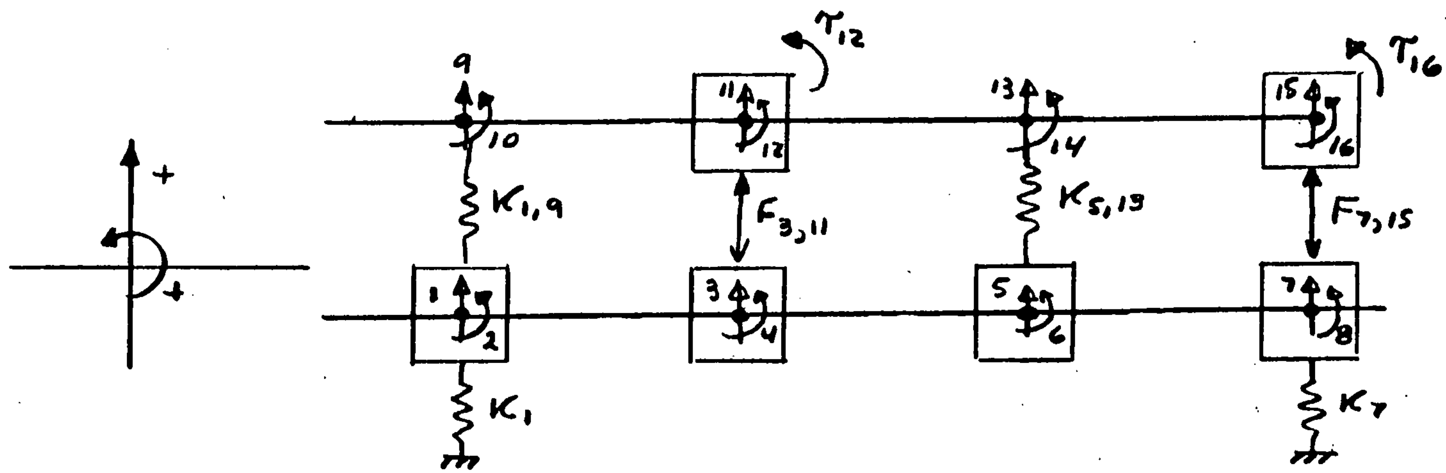


Figure 31. Root Locus for a Jeffcott Rotor With an Annular Seal and Casing Motion



PLANAR MODAL EQUATIONS

$$\begin{Bmatrix} \dot{\theta}_{12} \\ \theta_{16} \\ X_{11}-X_3 \\ X_{15}-X_7 \end{Bmatrix} = \frac{1}{S^2+\omega_k^2} \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} \begin{matrix} \tau_{12} \\ \tau_{16} \\ F_{3,11} \\ F_{7,15} \end{matrix}$$

COUPLING ELEMENTS

$$-\tau_{12} = [J_{12}S - J_{12}\Omega i]\,\dot{\theta}_{12} = H_1\,\theta_{12}$$

$$-\tau_{16} = [J_{16}S - J_{16}\Omega i]\,\dot{\theta}_{16} = H_2\,\theta_{16}$$

$$-F_{3,11} = [K_{11} + D_{11}S - B_{11}i](X_{11}-X_3) = H_3\,(X_{11}-X_3)$$

$$-F_{7,15} = [K_{15} + D_{15}S - B_{15}i](X_{15}-X_7) = H_4(X_{15}-X_7)$$

Figure 32. Planar Model for an Overhung Rotor

7.4 EXTENSION OF THE ROOT LOCUS TO A TYPICAL OVERHUNG ROTOR

The result of a number of approaches led to a very significant simplification in studies of rotor stability. The key is to assume that gyroscopic stiffening and seal damping with its resulting cross coupled term has no significant affect on mode shape, and that modal damping is small so that the resonant structural gain is adequately defined by considering individual modes and ignoring residual effects from other modes.

With these assumptions, the planar (no cross coupling) modes are computed for various speeds by including the speed-dependent direct stiffness at seals. Modal damping is assumed. The characteristic equation for an individual mode can then be written as:

$$M[S^2 + 2\zeta\omega S + \omega^2]\, X_m = \sum_n \Psi m\, \Psi n\, Fn$$

Angulation and deflection are defined in normalized form by mode shape, ϕm. As usual, the modal mass, M, is defined with the largest deflection being unity. Inputs, Fn, can be either forces or torques and the deflection, ϕn, at each input is required.

The cross coupling equations shown in Fig. 32 are now included. In terms of the system of Fig. 30, the equation of motion becomes:

$$S^2 + [2\zeta\omega + \frac{D_{11}}{M}(\phi_{11}-\phi_3)^2 + \frac{D_{15}}{M}(\phi_{15}-\phi_7)^2 - (J_{12}\phi_{12}^2-J_{16}\phi_{16}^2)\Omega i]S +$$
$$[\omega^2 - (B_{11}(\phi_{11}-\phi_3)^2 + B_{15}(\phi_{15}\phi_7)^2)\ i/2M] = 0$$

For purposes of demonstration we can separate real and imaginery coefficients and write:

$$S^2 + (R_1 - I_1 i)S + (\omega^2 - I_2 i) = 0$$

In real turbomachinery we find that the direct stiffness, damping and cross coupled stiffness are speed related. If values are obtained at reference speed, Ω, the values at another speed are:

$$K = \bar{K}\ (\Omega/\bar{\Omega})^2$$

$$D = \bar{D}\ (\Omega/\bar{\Omega})$$

$$B = \bar{D}\Omega/2 = D\ \Omega\ (\Omega/\bar{\Omega})^2/2$$

Therefore,

$$R_1 = 2\zeta\omega + [\bar{D}_{11}(\phi_{11}-\phi_3)^2 + \bar{D}_{15}(\phi_{15}-\phi_7)^2](\Omega/\bar{\Omega})/M$$

$$I_1 = (J_{12}\phi_{12}^2 + J_{16}\phi_{16}^2)\Omega$$

$$I_2 = [\bar{D}_{11}(\phi_{11}-\phi_3)^2 + \bar{D}_{15} - (\phi_{15}-\phi_7)^2]\bar{\Omega}\ (\Omega/\bar{\Omega})^2/2M$$

Speed-dependent stiffness is added to the zero-speed stiffness matrix prior to evaluation of the eigenvalue problem.

The cross coupling and damping effects on the system roots associated with each mode are then determined by solving the quadratic in S where the coefficients are complex. Physical systems where quadratics with complex coefficients arise are unusual and perhaps unique in engineering to rotating machinery.

To demonstrate the effect of these complex coefficients, we can consider two problems: one where $I_1 = 0$ and one where $I_2 = 0$. First, where $I_1 \neq 0$, $I_2 = 0$, we have a rotating system with gyroscopic cross coupling. Assuming ζ and I_1/ω are small, solutions for S are:

$$S_1 \approx -\zeta\omega\ (1+I_1/2\omega) + i\omega\ (1+I_1/2\omega)$$

$$S_2 \approx -\zeta\omega\ (1-I_1/2\omega) - i\omega\ (1-I_1/2\omega)$$

The sign of the imaginery coefficient indicates forward (+) or backward (-) precession relative to shaft rotation. The more negative the real coefficient, the more the damping losses. Thus, gyroscopic forces increase the frequency and damping of forward precession while reducing the frequency and damping of backward precession. For zero system damping (ζ), gyroscopic forces do not cause instability but simply change precession frequencies.

The case where $I_1 = 0$, $I_2 \neq 0$ can also be examined for small values of I_1/ω^2. This results in:

$$S_1 \approx -\zeta\omega + I_2/2\omega^2 + i\ \omega$$

$$S_2 \approx -\zeta\omega - I_2/2\omega^2 - i\ \omega$$

In this case, the frequency is not affected, but the damping of forward precession can be driven to negative values. Backward precession has increased damping.

For a rotor-casing model with all the cross coupling terms, as speed is increased one would expect all the planar modal frequencies to increase due to nonlinear stiffening of the direct spring rate. Gyroscopic forces would then cause the forward precession frequency to increase slightly while the backward precession frequency decreases slightly. Dependent on mode shape, the damping of forward precession could increase due to gyroscopic forces or decrease due to seal coefficients. The effect on the backward precession component should be opposite that of forward precession. As speed increases one would finally expect the seal forces to dominate (since they vary with speed squared) and negative damping to prevail.

In order to evaluate this approach, a FORTRAN program was written for the IBM-PC. A program, ROTOR2, was written with five subroutines. Listings for these programs are contained in Appendix E. The program also calls two scratch data files which contain data for the last system run with the program. They are contained on the disk with the executable program file. The program is run using the CRT display for user interrogation, while significant output is diverted to the printer. During interrogation, the program has been written so that no zero values for geometric values are allowed. This simplifies logic and computer setup

and maximizes running speed for best interactive performance. Using an 8087 chip, setup requires about 3 minutes while each speed iteration requires 30 seconds (10 seconds of processing and 20 seconds of printing) for 4 modes of interest.

A printed output for a session is also contained in Appendix E. The data inputs (which are called to the CRT screen) are indicated. After the frequencies for the planar modes are printed, the program asks how many modes the user would like to track in the speed increment loop. Usually, only those less than the potential speed range are of interest, but all 16 modes may be tracked with an additional penalty due to printing.

The output shows the significance of relative motion between the case and rotor. Modes which show high relative displacement at the seal connection points show the most tendency to become unstable due to cross coupling. It is desirable to have such modes appear at frequencies greater than half the running speed to avoid degradation of inherent modal damping.

The tendency for half speed whirl is vividly displayed using this approach by considering the equation for real forces.

$$-F = [K + DS - i\ D\Omega/2]\Delta X$$

For a modal frequency where $S = i\omega$, all damping from the seal vanishes at $\Omega = 2\omega$ and becomes negative for higher speeds. At higher speeds, negative damping from the seal will eventually overcome inherent modal damping and result in instability which will occur at a frequency of ω. A slight increase in the whirl speed (due to gyroscopic forces) and the violence of the instability (limited by physical nonlinearities) is observed in data as speed is increased past the stable point.

8.0 RESULTS AND OBSERVATIONS

The results of this study are significant in several areas and exceed the planned scope of the work in some ways.

The initial phase produced a summary of a number of high speed rotors and their properties. While not all of the existing pumps were applicable to the type of study being performed, several of them were. Where analytical models were available, properties such as component weight and system frequencies were obtained for reference. Results of some of the stability analyses for these units are included to demonstrate the stability activity with speed. In one case, upper modes were deleted to determine the affect on computed stability and it was found that severe truncation had little affect on the stability results.

An additional step used a simplified generic model of a rotor mounted through bearing spring rates to ground in comparison with one where the bearing springs were connected to a casing which was attached with unsymmetrical springs to ground. Casing motion increased the speed at which instability occurred. The clue to increased stability in this case may be the lack of symmetry of the casing supports. Some investigations have indicated that unsymmetrical bearing spring rates attached to ground also improve the stable speed range. This aspect of rotor dynamics was not pursued in this study.

In order to have a tool to allow valid extensive models of turbopumps to be run without step-by-step batch-processing, a number of mainframe codes were collected and revised for use in an IBM-PC. This allows complex pump-housing configurations to be run in a much shorter schedule and in an interactive mode. The code has all the accuracy and capability of the summation of the larger programs and is written in FORTRAN. Some sample programs and a user guide are included in this report. An interesting and useful output is the roots in the speed range of the pump plotted in the complex plane as loci of speed. The plot indicates the change in frequency and closed loop damping as speed is increased. Troublesome system resonant modes can be easily tracked. The affect on these roots of system changes are easily seen.

During the course of this study, a more simplified approach to examining turbopump stability was a goal. After a number of approaches to this part of the study, it was found that an extremely simple concept could be developed by assuming (1) a symmetric structural system, and (2) that including gyroscopic forces and seal damping and cross coupling did not significantly affect mode shapes. The first assumption leads to the transformations $Y = -iX$ and $F_y = -iF_x$ allows evaluation of the dynamics in one plane followed by application of cross coupled forces. The second assumption allows cross coupling effects on a mode to be evaluated by determining the roots of a quadratic with complex coefficients.

Examining the equation for a seal with cross coupling, using the concept that $Y = -ix$ the seal force becomes

$$-F = [K + DS - Di\Omega/2]\Delta X$$

For a particular mode where $s = i\omega$ we find that the imaginary coefficient (the damping force) becomes negative for speeds greater than twice the modal frequency. At some greater speed this negative damping will exceed inherent damping in the system, causing unstable precession at less than half speed. This only occurs for forward precession. The backward precession component (where $S = -i\omega$) receives increased damping as speed is increased.

Gyroscopic forces act as decreased mass in forward precession increasing the frequency and as increased mass in backward precession decreasing the frequency. The damping is increased for forward precession and decreased by backward precession. Gyroscopic forces can cause instability in backward precession if the mode has inherent damping. A conservative mode, however, will not be destabilized by small gyroscopic forces.

A generic rotor program was written in FORTRAN for the IBM-PC based on the simplifications. These show the trends expected although the results have not been checked numerically with a full model. The results indicate that modes with high relative deflection between the case and rotor at seals are most susceptible to instability if the frequency is less than half the operating shaft speed.

9.0 SUGGESTIONS FOR ADDITIONAL EFFORT

Several areas of additional effort are indicated.

First, it would be advisable to numerically compare the simplified rotor model with results obtained with the complete model RSTAB.

A second area of activity would be to use the complex quadratic type solution for single modes, but including a more realistic geometry; i.e., nonuniform rotor stiffness, a more complex casing model, etc., to allow definition of more elements with cross coupling. The method appears to be quite general for axisymmetric systems.

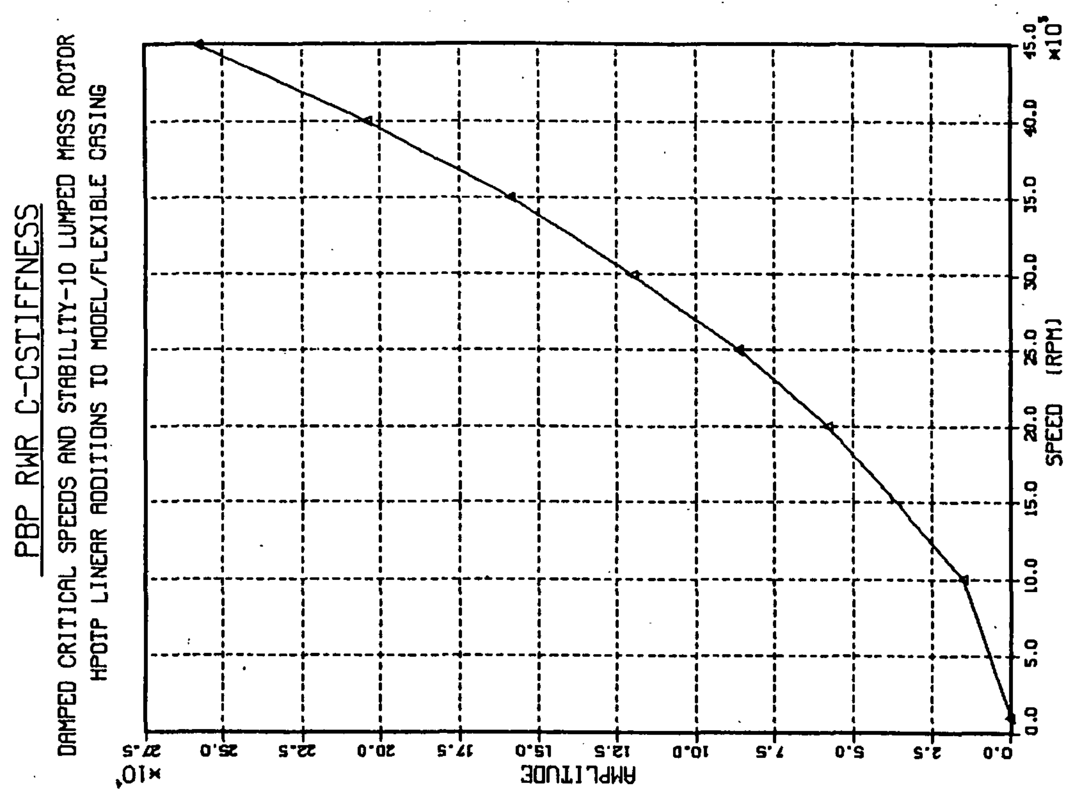
The third area where additional effort in the Root Locus vein may be significant is for a casing with asymmetrical stiffness. While this complexity does not allow direct application of the concept $Y = -iX$, it may be possible to generalize the problem into polar coordinates in a similar manner using one mode in each of two planes to estimate the dynamic affect. Elliptical motion is obviously expected. but the approach might pay large dividends in basic knowledge.

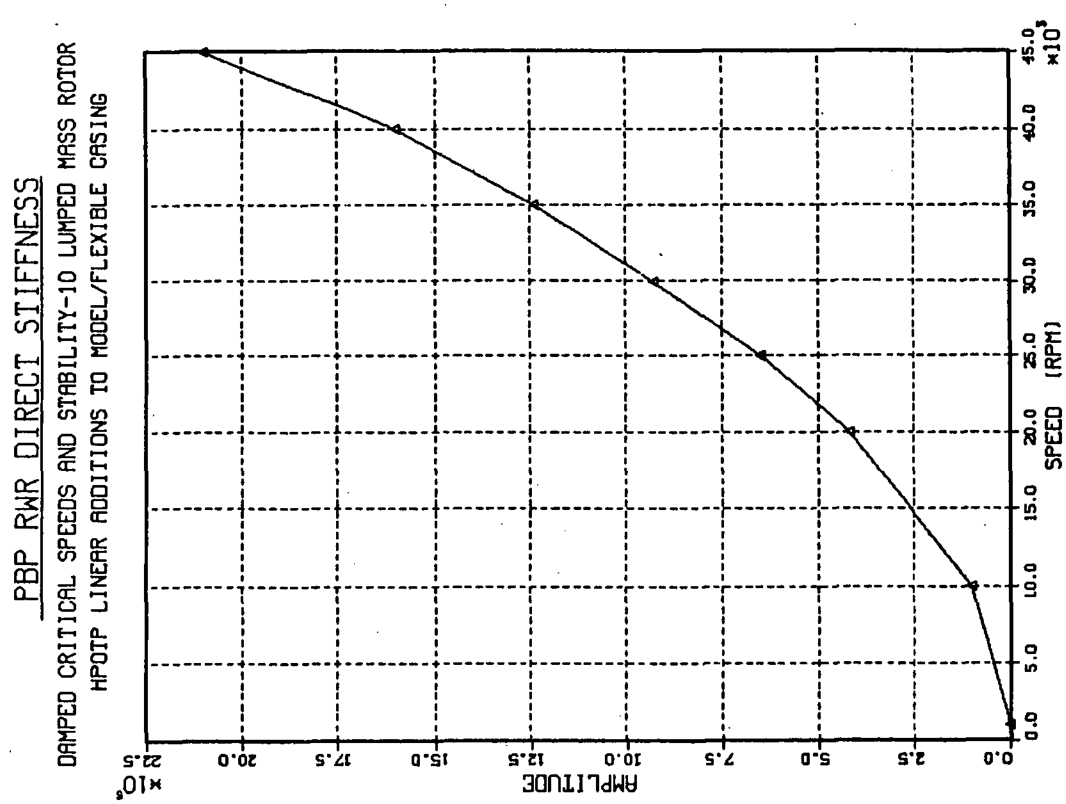
NOMENCLATURE

D	Damping coefficient
E	Elastic modulus
F	Force
f	Frequency (Hz)
H	Horsepower
h	Turbine blade height
i	-1
I	Structural section modulus
J	Mass moment of inertia about Y or Z axis
Jp	Polar moment of inertia
K	Stiffness coefficient
L	Length
M	Mass
M(n)	Mass associated with eigenvector n
N	Shaft speed (RPM)
P	Pressure
Q	Dynamic pressure
q	Eigenvector
R	Radius
S	Laplace operator
t	Time
V	Velocity
X,Y,Z	Coordinate system
W	Weight
α	Density
δ	Relative case-rotor deflection
ε	Radial clearance
ξ	Damping factor (fraction of critical)
θ	Angulation about Z axis
λ	Eigenvalue
μ	Viscosity
ρ	Radial distance
σ	Viscous pressure drop Q
Φ	Normalized modal displacement
ψ	Shaft speed (rad/sec)
ω	Frequency (rad/sec)
ω	Frequency associated with eigenvalue
Ω	Shaft speed (rad/sec)

APPENDIX A

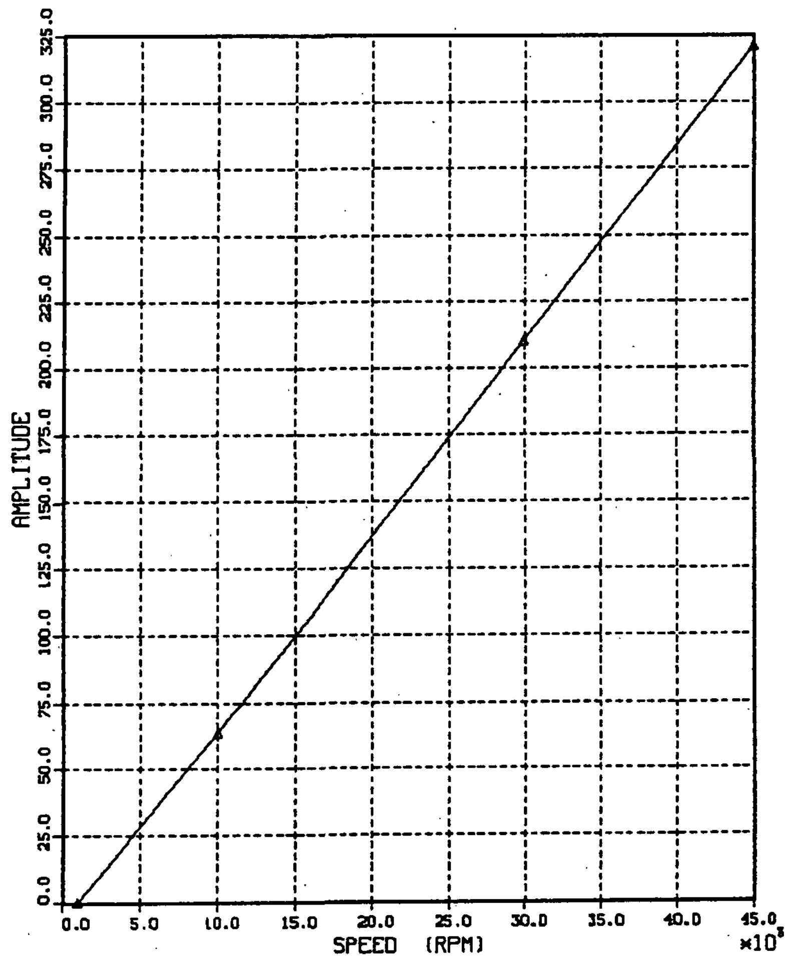
SUMMARY OF PHYSICAL PROPERTIES OF A NUMBER OF EXISTING HIGH-SPEED ROTORS


PBP RWR C-CSTIFFNESS
DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING
AMPLITUDE
x10^4
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
27.5
SPEED (RPM)
x10^3
0.0
5.0
10.0
15.0
20.0
25.0
30.0
35.0
40.0
45.0


PBP RWR DIRECT STIFFNESS
DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING
AMPLITUDE
x10^5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
SPEED (RPM)
x10^3
0.0
5.0
10.0
15.0
20.0
25.0
30.0
35.0
40.0
45.0

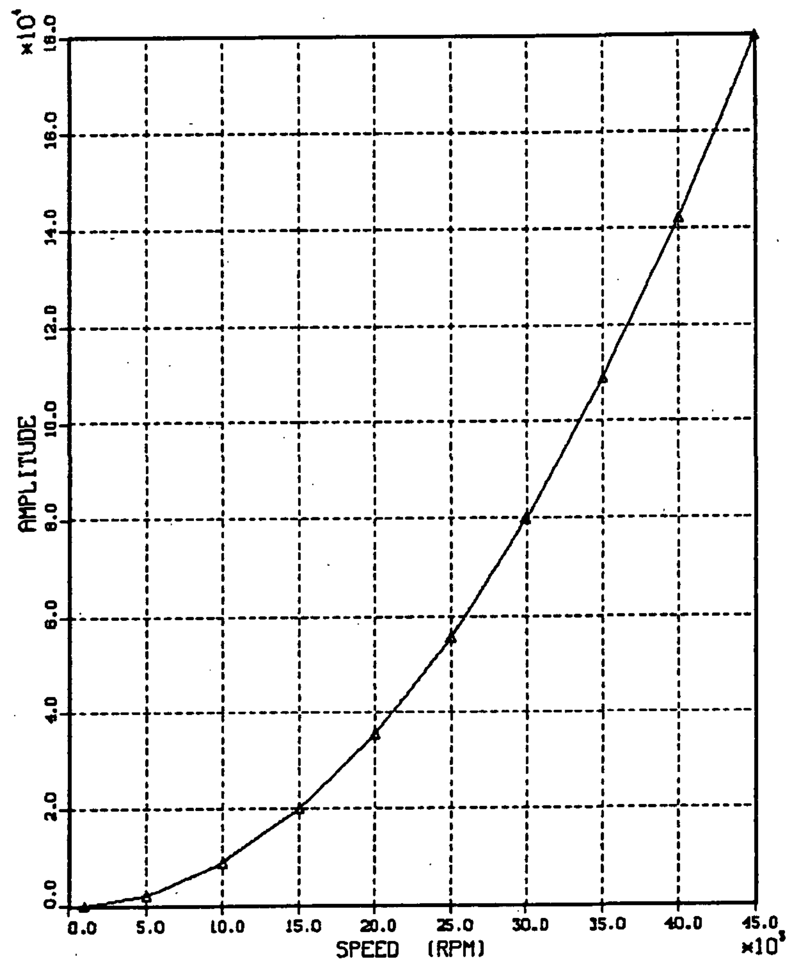
PBP RWR DIRECT DAMPING

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING



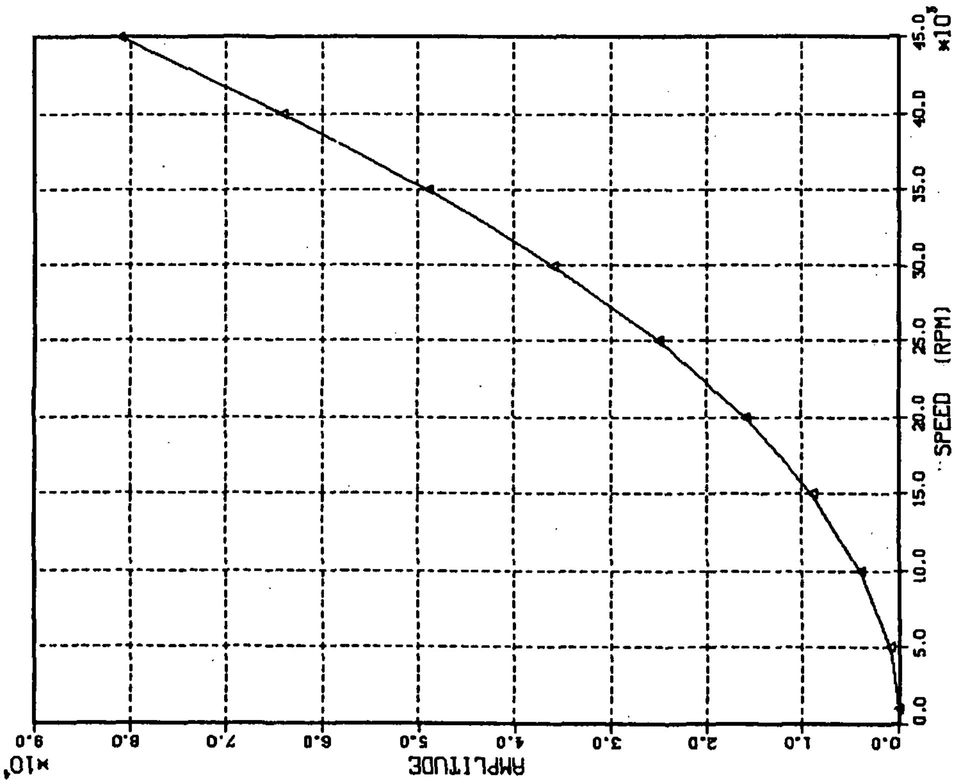
M IMP DIRECT STIFFNESS

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING



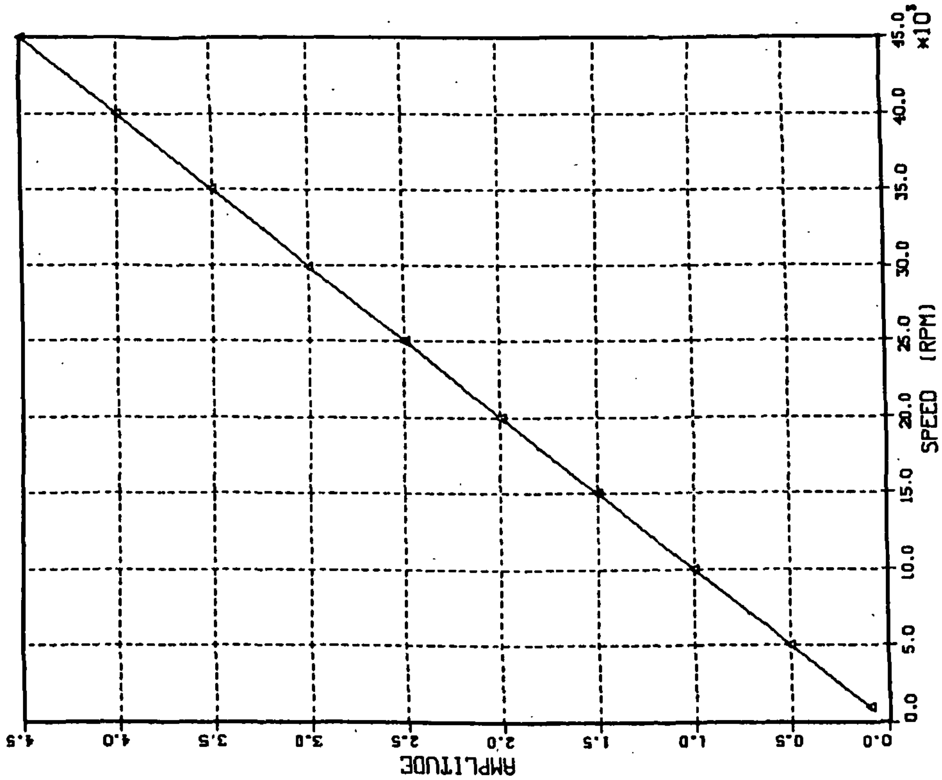
M IMP C-C STIFFNESS

DAMPED CRITICAL SPEEDS AND STABILITY-1D LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING



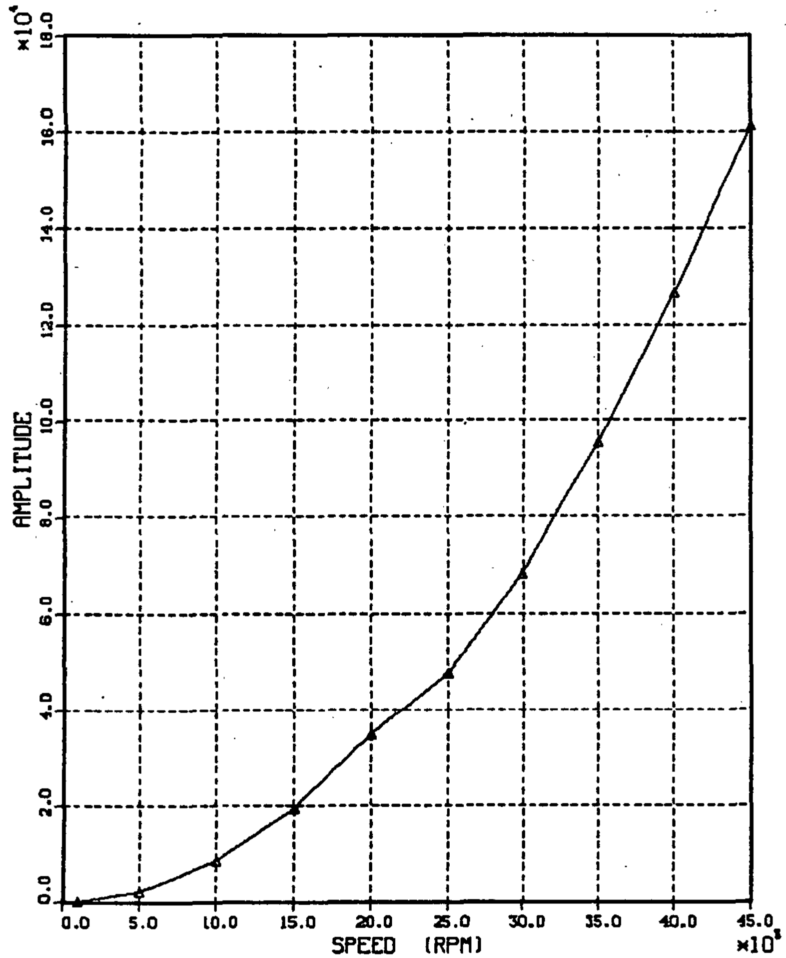
M IMP DIRECT DAMPING

DAMPED CRITICAL SPEEDS AND STABILITY-1D LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING



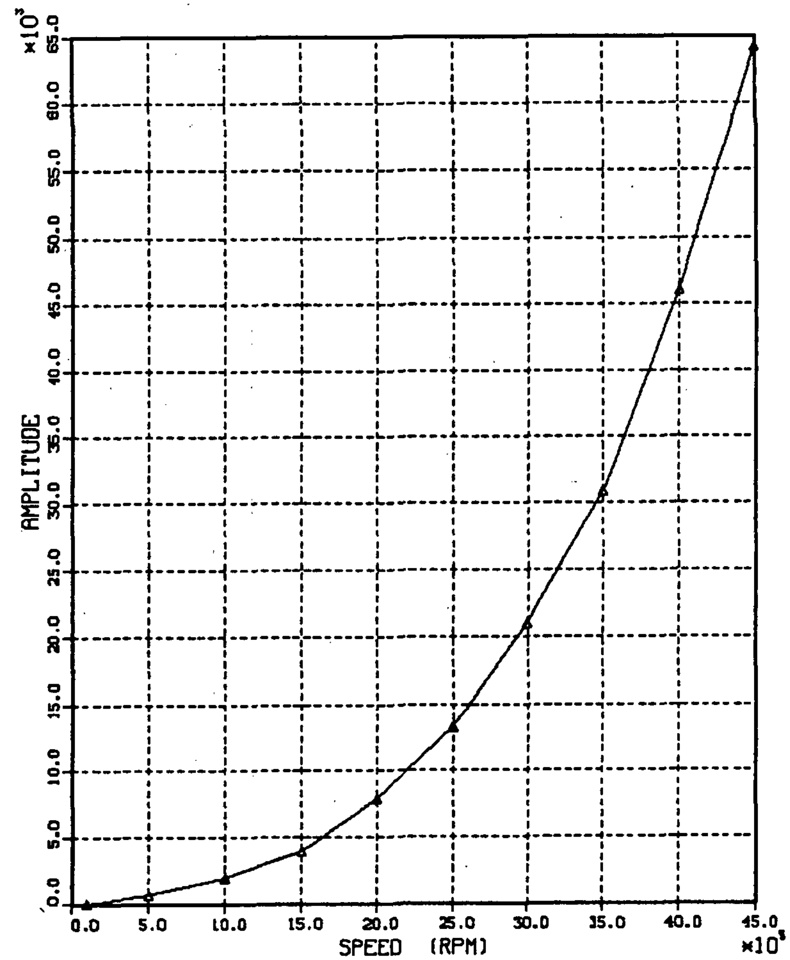
TURBINE DIRECT STIFFNESS

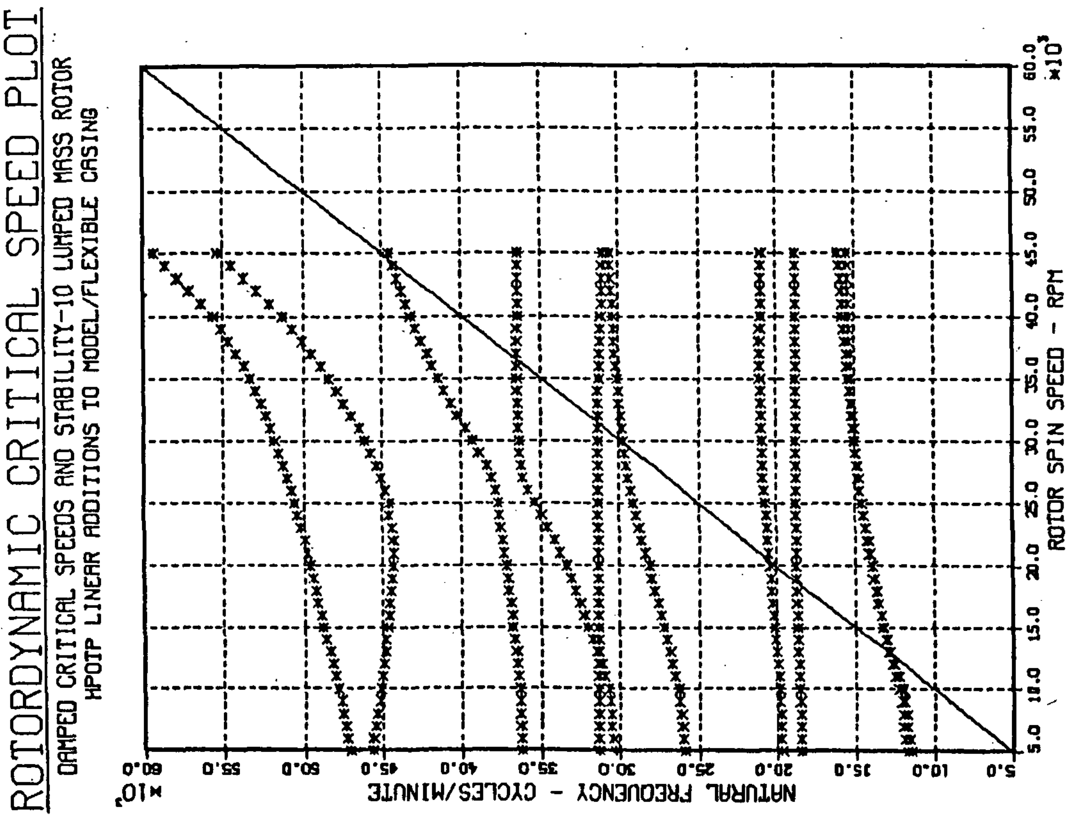
DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING

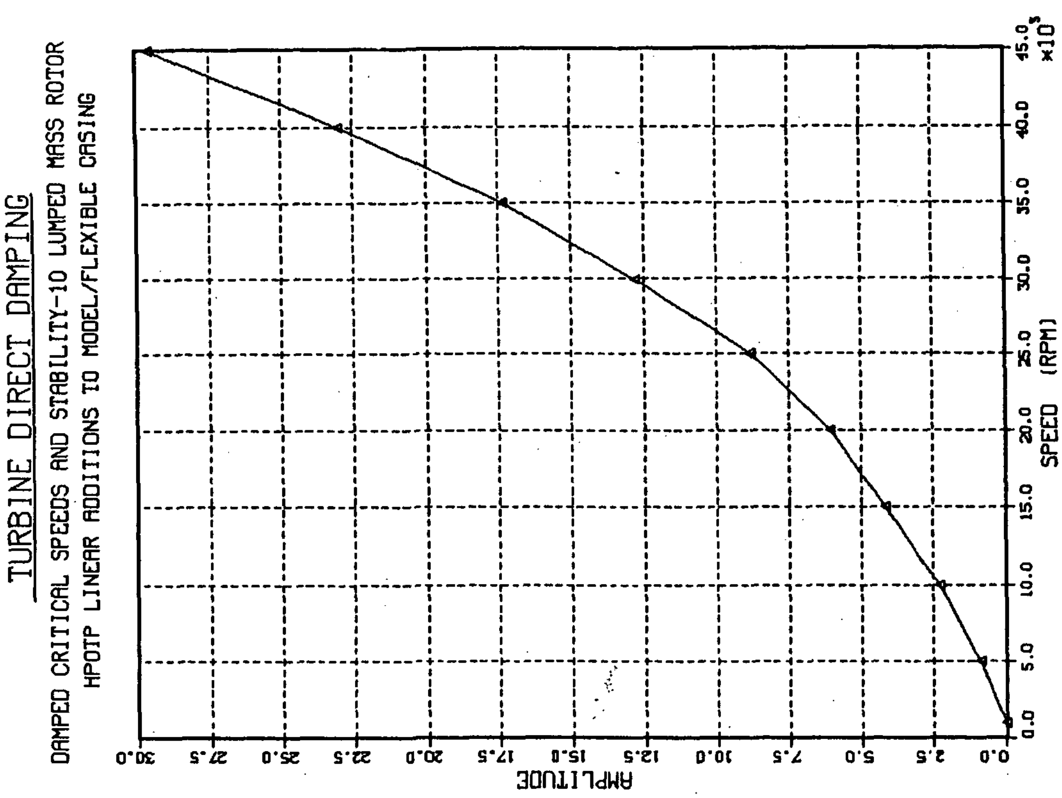


TURBINE C-C STIFFNESS

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING

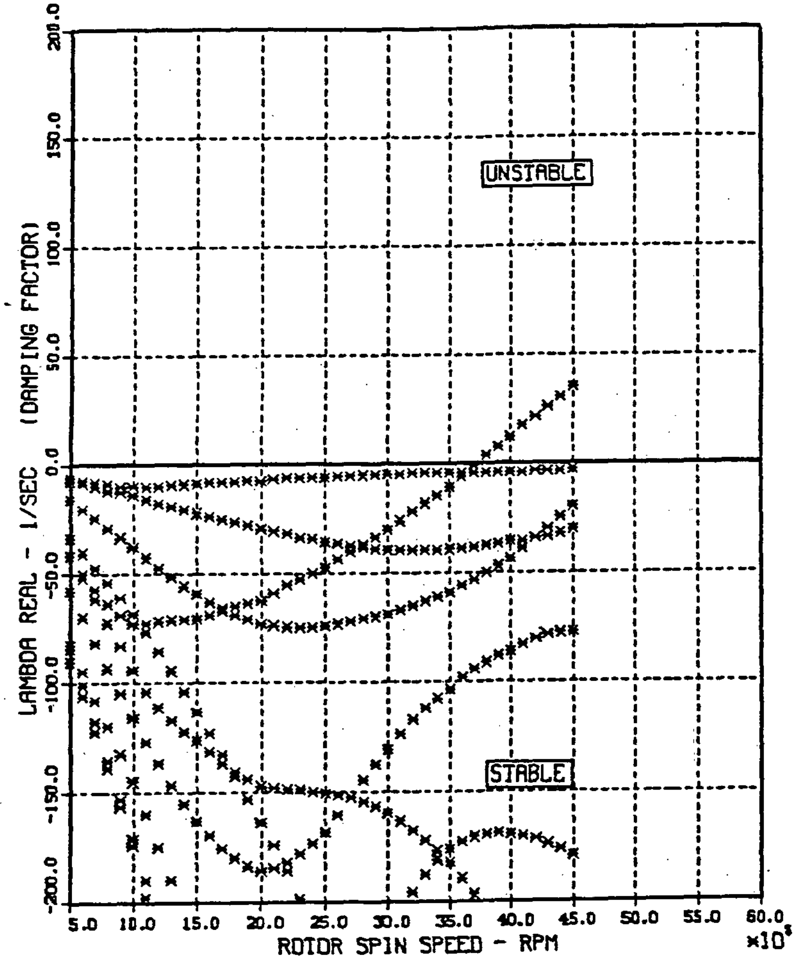



ROTORDYNAMIC CRITICAL SPEED PLOT
DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING
NATURAL FREQUENCY - CYCLES/MINUTE
×10³
ROTOR SPIN SPEED - RPM
×10³


TURBINE DIRECT DAMPING
DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING
AMPLITUDE
SPEED (RPM)
×10³

ROTORDYNAMIC STABILITY PLOT

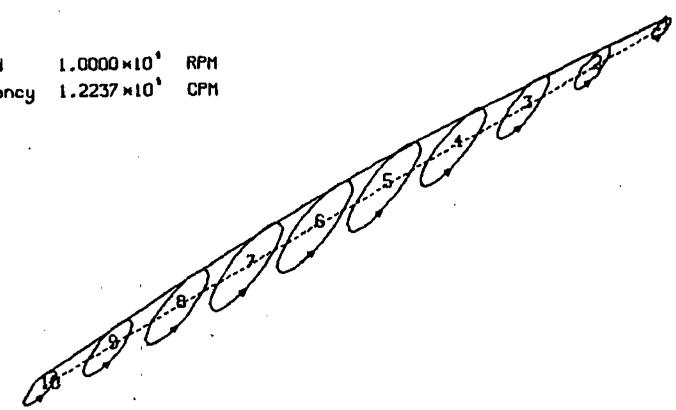
DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING



Rotor Group Mode Shape

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING

Spin Speed 1.0000×10^4 RPM
Nat Frequency 1.2237×10^4 CPM

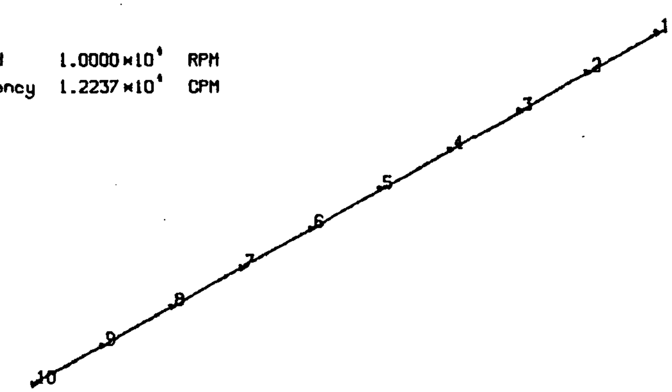


STATION 1 ORBIT - FORWARD PRECESSION

Casing Group Mode Shape

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING

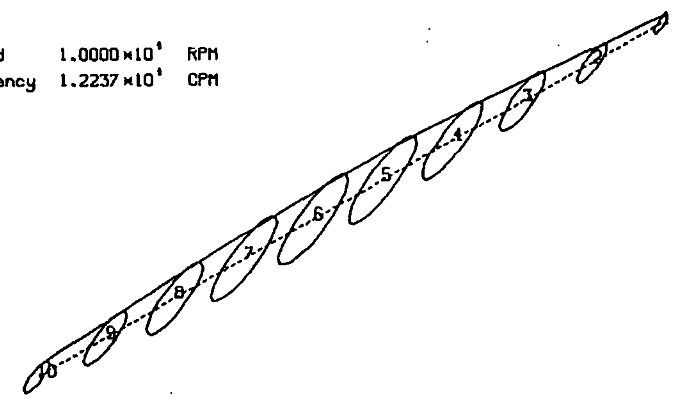
Spin Speed 1.0000×10[4] RPM
Nat Frequency 1.2237×10[4] CPM



Relative Deflection Mode Shape

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING

Spin Speed 1.0000×10[4] RPM
Nat Frequency 1.2237×10[4] CPM



STATION 1 ORBIT - FORWARD PRECESSION

APPENDIX B

USE OF MATRIX METHODS AND SUBSTRUCTURE MODELING IN ROTOR STABILITY ANALYSIS

This technique allows independently formulated component (rotor and casing) models, which should be reduced in degrees of freedom via a simplified modeling approach or a Guyan reduction, to be combined in normal coordinates into a coupled system representing a generic turbopump. Interaction between the substructures, as well as general stiffness and damping modifications internal to the substructure components, may be specified in the physical (or original) coordinate systems and incorporated with the system by proper transformation into the coupled system normal coordinate system.

The coupled system is then solved for its complex eigenvalues and eigenvectors, which yield the required information regarding the stability margins of the system modes and, via backtransformation to the physical coordinate system, the direction of precession of the rotor degrees of freedom. By permitting the specification of rpm-dependency of each system addition and modification, the behavior of generic turbopump interaction elements, such as bearings, seals and turbines, can be accurately modeled. The eigenvalue solution process is then repeated while parametrically varying the pump speed.

The resulting information is presented in the form of critical speeds (from the imaginary part of the complex eigenvalue) and modal stability margins (from the real part of the complex eigenvalue) versus pump speed. A root locus diagram is made, which combines the information contained in the critical speed and stability plots. Backtransformation and plotting of the coupled system displacement mode shapes also is performed.

The analytical method (Ref. 1) begins with normalization of the system eigenvectors with respect to the mass matrix such that:

$$X^T M X = M_N = I;$$

Ref. 1. Timoshenko, S., D. H. Young, and W. Weaver, Jr., "Vibration Problems in Engineering," Fourth Edition, John Wiley and Sons, 1974, pp 296-303.

where

X = the eigenvector matrix (T denotes transposition),

M = the mass matrix (in physical coordinates),

M_N = normal mass matrix, and

I = the identity matrix

It then follows by similar orthogonal transformation that:

$$S_N = \begin{bmatrix} \omega_1^2 & & & \\ & \omega_2^2 & & \\ & & \ddots & \\ & & & \omega_n^2 \end{bmatrix}, \text{ and } D_N = \begin{bmatrix} 2\omega_1\xi_1 & & & \\ & 2\omega_2\xi_2 & & \\ & & \ddots & \\ & & & 2\omega_n\xi_n \end{bmatrix};$$

where:

S_N = the normal stiffness matrix,

D_N = the normal damping matrix,

ω_i = the natural frequency of the ith normal mode, and

ξ_i = the modal damping ratio of the ith normal mode.

The normalized system equations of motion may then be written as:

$$I\ddot{\mu} + D_N\dot{\mu} + S_N\mu = 0;$$

where:

μ = the normal coordinate displacement vector.

The substructure interaction elements and general modifications are specified in physical coordinates and transformed via the normalized eigenvectors into the system normal coordinates. For example:

$$X^T \Delta SX = \Delta S_N, \text{ and}$$

$$S_N' = S_N + \Delta S_N;$$

where

ΔS = the stiffness addition/modification matrix in physical coordinates

ΔS_N = the stiffness addition/modification matrix in system normal coordinates, and

S_N' = the new system stiffness matrix

The new system equations of motion are then written in state-variable form:

$$\left[\begin{array}{c|c} I & \emptyset \\ \hline \emptyset & I \end{array}\right] \left\{\begin{array}{c} \ddot{\mu} \\ \hline \dot{\mu} \end{array}\right\} + \left[\begin{array}{c|c} C_N' & S_N' \\ \hline -I & \emptyset \end{array}\right] \left\{\begin{array}{c} \dot{\mu} \\ \hline \mu \end{array}\right\} = 0,$$

which is then solved by an eigenvalue solution routine which solves the general nonsymmetrical dynamical matrix (IMSL routine EIGRF is being used for this purpose).

Critical speed and modal stability data are obtained from the complex eigenvalues computed at each rpm step in the analysis. These are computed in the form:

$$\omega_j = a_j + ib_j \quad \text{for} \quad j = 1, N;$$

where

ω_j = $j\underline{th}$ coupled system complex eigenvalue,

a_j = real component of ω_j

b_j = imaginary component of ω_j,

i = $\sqrt{-1}$, and

N = total coupled system degrees of freedom in normal coordinates (number of casing substructure modes plus number of rotor substructure modes)

The a_j represents the exponential decay (or growth, for $a_j > o$) of the jth mode when disturbed from its equilibrium position. When the a_j component is plotted versus pump speed, the result is a stability margin diagram. The mode is stable when a_j is less than zero, and unstable when a_j is greater than zero.

The b_j represents the damped natural frequency of the jth mode. When b_j component is plotted versus pump speed, the result is a critical speed map. The intersection of the b_j curve with the diagonal line indicating synchronous speed indicates a potential critical speed. The combination of the modal stability and critical speed plots is the more familiar root locus plot, where the a_j and b_j are plotted together on the imaginary plane. In this case, the pump speed is represented parametrically as the locus of the complex roots.

The complex system eigenvectors can be backtransformed to the physical coordinate system using the original normalized substructure mode shapes:

$$X' = XZ;$$

where

X' = the casing-rotor coupled system modal displacement vector matrix in physical coordinates,

X = the original normalized substructure modal matrix, and

Z = the complex system eigenvector matrix computed for each rpm step

The coupled system modal displacement vector is then printed/plotted to aid in interpretation of its significance to rotor stability and bearing loads.

Immediate indications of strong casing-rotor interaction are available from the coupled system mode shapes. Modes which exhibit comparable motion of both casing and rotor degrees of freedom will be most significant in assessing the influence of casing flexibility on rotor dynamics. Large relative motion between casing and rotor degrees of freedom representing a bearing coupling element inicates a potentially significant contributor to bearing loads.

Using vector analysis methods, the direction of rotor precession is a simple calculation involving the complex translational degrees of freedom (y, z) at a rotor longitudinal "station." Evaluated at any time, t (t = o is convenient), displacement and velocity vectors are defined as (Fig. B-1):

$$R = Re[y]\hat{j} + Re[z]\hat{k}; \text{ and}$$
$$\dot{R} = Re[\dot{y}]\hat{j} + Re[\dot{z}]\hat{k};$$

$$\bar{R} = Re[y]\,\hat{j} + Re[z]\,\hat{k}$$

$$\dot{\bar{R}} = Re[\dot{y}]\,\hat{j} + Re[\dot{z}]\,\hat{k}$$

$$\dot{\bar{p}} = \bar{R} \times \dot{\bar{R}}$$

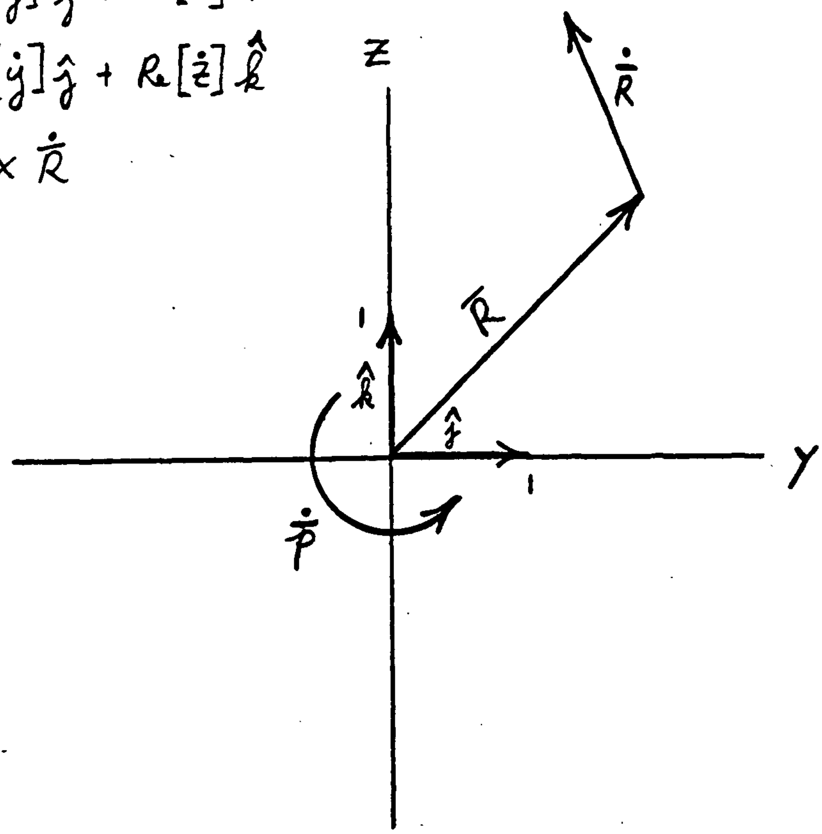


Figure B-1. Calculation of Rotor Precession From the Complex Displacement Mode Shape

where

Re[ ] = the real component of the complex argument, and

$\hat{j}$, $\hat{k}$ = unit vectors aligned with the y and z axes, respectively

It then follows by vector cross-product, that the precession vector, $\dot{\bar{p}}$, is:

$$\dot{\bar{p}} = \bar{R} \times \dot{\bar{R}} = (Re[y]\ Re\ [\dot{z}] - Re[z]\ Re[\dot{y}])\hat{i}$$

where

$\hat{i}$ = the unit vector aligned with the x (longitudinal) axis

The modal displacement and velocity quantities are related according to:

$$\dot{y} = \lambda y, \text{ and } \dot{z} = \lambda z;$$

where:

λ = the complex eigenvalue.

Substitution of these into the vector cross-product results (after some manipulation) in:

$$\dot{\bar{p}} = \mathrm{Im}[\lambda]\ (\mathrm{Re}[z]\ \mathrm{Im}[y] - \mathrm{Re}[y]\ \mathrm{Im}[z])\hat{i};$$

where

Im[ ] = the imaginary component of the complex argument.

This equation is easily evaluated at each rotor station.

APPENDIX C

PROGRAM RSTAB OUTPUT LISTING FOR PROGRAM VERIFICATION TEST CASE

```
***********************************************************************************
*****                                                                         *****
*****                    P R O G R A M   R S T A B                            *****
*****                                                                         *****
 ****                    IBM PC VERSION 1.0                                   *****
*****                                                                         *****
***********************************************************************************
```

```
* *  DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR FLEX CASE     * *

    NO. OF ROTOR DOF   =  40     NO. OF ROTOR MODES  =   6

    NO. OF CASING DOF  =  40     NO. OF CASING MODES =   6

    ROTOR TAPE CODE    =   1     CASING TAPE CODE    =   1

    NO. OF GYROSCOPIC ADDITIONS =  20

    ACCELERATION OF GRAVITY     =  386.40
```

* * * ROTOR GROUP MODAL MATRIX * * *

MODE NO. 1

3.81387E+00	.00000E+00	.00000E+00	-1.52437E-01	3.35656E+00
.00000E+00	.00000E+00	-1.52437E-01	2.89925E+00	.00000E+00
.00000E+00	-1.52437E-01	2.44194E+00	.00000E+00	.00000E+00
-1.52437E-01	1.98462E+00	.00000E+00	.00000E+00	-1.52437E-01
1.52731E+00	.00000E+00	.00000E+00	-1.52437E-01	1.07000E+00
.00000E+00	.00000E+00	-1.52437E-01	6.12687E-01	.00000E+00
.00000E+00	-1.52437E-01	1.55375E-01	.00000E+00	.00000E+00
-1.52437E-01	-3.01937E-01	.00000E+00	.00000E+00	-1.52437E-01

MODE NO. 2

.00000E+00	3.81387E+00	1.52437E-01	.00000E+00	.00000E+00
3.35656E+00	1.52437E-01	.00000E+00	.00000E+00	2.89925E+00
1.52437E-01	.00000E+00	.00000E+00	2.44194E+00	1.52437E-01
.00000E+00	.00000E+00	1.98462E+00	1.52437E-01	.00000E+00
.00000E+00	1.52731E+00	1.52437E-01	.00000E+00	.00000E+00
1.07000E+00	1.52437E-01	.00000E+00	.00000E+00	6.12687E-01
1.52437E-01	.00000E+00	.00000E+00	1.55375E-01	1.52437E-01
.00000E+00	.00000E+00	-3.01937E-01	1.52437E-01	.00000E+00

MODE NO. 3

-1.67022E+00	.00000E+00	.00000E+00	1.91526E-01	-1.09565E+00
.00000E+00	.00000E+00	1.91526E-01	-5.21070E-01	.00000E+00
.00000E+00	1.91526E-01	5.35071E-02	.00000E+00	.00000E+00
1.91526E-01	6.28084E-01	.00000E+00	.00000E+00	1.91526E-01
1.20266E+00	.00000E+00	.00000E+00	1.91526E-01	1.77724E+00
.00000E+00	.00000E+00	1.91526E-01	2.35182E+00	.00000E+00
.00000E+00	1.91526E-01	2.92639E+00	.00000E+00	.00000E+00
1.91526E-01	3.50097E+00	.00000E+00	.00000E+00	1.91526E-01

MODE NO. 4

.00000E+00	-1.67022E+00	-1.91526E-01	.00000E+00	.00000E+00
-1.09565E+00	-1.91526E-01	.00000E+00	.00000E+00	-5.21070E-01
-1.91526E-01	.00000E+00	.00000E+00	5.35071E-02	-1.91526E-01
.00000E+00	.00000E+00	6.28084E-01	-1.91526E-01	.00000E+00
.00000E+00	1.20266E+00	-1.91526E-01	.00000E+00	.00000E+00
1.77724E+00	-1.91526E-01	.00000E+00	.00000E+00	2.35182E+00
-1.91526E-01	.00000E+00	.00000E+00	2.92639E+00	-1.91526E-01
.00000E+00	.00000E+00	3.50097E+00	-1.91526E-01	.00000E+00

MODE NO. 5

-4.23564E+00	.00000E+00	.00000E+00	6.21006E-01	-2.37190E+00
.00000E+00	.00000E+00	6.06796E-01	-5.89780E-01	.00000E+00
.00000E+00	5.50129E-01	9.20328E-01	.00000E+00	.00000E+00
4.14930E-01	1.91305E+00	.00000E+00	.00000E+00	2.11594E-01
2.20052E+00	.00000E+00	.00000E+00	-4.19694E-02	1.66307E+00
.00000E+00	.00000E+00	-2.87072E-01	4.65316E-01	.00000E+00
.00000E+00	-4.70629E-01	-1.17521E+00	.00000E+00	.00000E+00
-5.79026E-01	-2.94536E+00	.00000E+00	.00000E+00	-5.90858E-01

MODE NO. 6

.00000E+00	-4.23564E+00	-6.21006E-01	.00000E+00	.00000E+00
-2.37190E+00	-6.06796E-01	.00000E+00	.00000E+00	-5.89780E-01
-5.50129E-01	.00000E+00	.00000E+00	9.20328E-01	-4.14930E-01
.00000E+00	.00000E+00	1.91305E+00	-2.11594E-01	.00000E+00
.00000E+00	2.20052E+00	4.19694E-02	.00000E+00	.00000E+00
1.66307E+00	2.87072E-01	.00000E+00	.00000E+00	4.65316E-01
4.70629E-01	.00000E+00	.00000E+00	-1.17521E+00	5.79026E-01
.00000E+00	.00000E+00	-2.94536E+00	5.90858E-01	.00000E+00

* * * CASING GROUP MODAL MATRIX * * *

MODE NO. 1

1.32586E+00	.00000E+00	.00000E+00	-6.06475E-02	1.14392E+00
.00000E+00	.00000E+00	-6.06475E-02	9.61978E-01	.00000E+00
.00000E+00	-6.06475E-02	7.80036E-01	.00000E+00	.00000E+00
-6.06475E-02	5.98093E-01	.00000E+00	.00000E+00	-6.06475E-02
4.16151E-01	.00000E+00	.00000E+00	-6.06475E-02	2.34208E-01
.00000E+00	.00000E+00	-6.06475E-02	5.22656E-02	.00000E+00
.00000E+00	-6.06475E-02	-1.29677E-01	.00000E+00	.00000E+00
-6.06475E-02	-3.11620E-01	.00000E+00	.00000E+00	-6.06475E-02

MODE NO. 2

.00000E+00	1.32586E+00	6.06475E-02	.00000E+00	.00000E+00
1.14392E+00	6.06475E-02	.00000E+00	.00000E+00	9.61978E-01
6.06475E-02	.00000E+00	.00000E+00	7.80036E-01	6.06475E-02
.00000E+00	.00000E+00	5.98093E-01	6.06475E-02	.00000E+00
.00000E+00	4.16151E-01	6.06475E-02	.00000E+00	.00000E+00
2.34208E-01	6.06475E-02	.00000E+00	.00000E+00	5.22656E-02
6.06475E-02	.00000E+00	.00000E+00	-1.29677E-01	6.06475E-02
.00000E+00	.00000E+00	-3.11620E-01	6.06475E-02	.00000E+00

MODE NO. 3

.00000E+00	6.96948E-02	5.03815E-02	.00000E+00	.00000E+00
-8.14498E-02	5.03815E-02	.00000E+00	.00000E+00	-2.32594E-01
5.03815E-02	.00000E+00	.00000E+00	-3.83739E-01	5.03815E-02
.00000E+00	.00000E+00	-5.34883E-01	5.03815E-02	.00000E+00
.00000E+00	-6.86028E-01	5.03815E-02	.00000E+00	.00000E+00
-8.37173E-01	5.03815E-02	.00000E+00	.00000E+00	-9.88317E-01
5.03815E-02	.00000E+00	.00000E+00	-1.13946E+00	5.03815E-02
.00000E+00	.00000E+00	-1.29061E+00	5.03815E-02	.00000E+00

MODE NO. 4

-6.96948E-02	.00000E+00	.00000E+00	5.03815E-02	8.14498E-02
.00000E+00	.00000E+00	5.03815E-02	2.32594E-01	.00000E+00
.00000E+00	5.03815E-02	3.83739E-01	.00000E+00	.00000E+00
5.03815E-02	5.34883E-01	.00000E+00	.00000E+00	5.03815E-02
6.86028E-01	.00000E+00	.00000E+00	5.03815E-02	8.37173E-01
.00000E+00	.00000E+00	5.03815E-02	9.88317E-01	.00000E+00
.00000E+00	5.03815E-02	1.13946E+00	.00000E+00	.00000E+00
5.03815E-02	1.29061E+00	.00000E+00	.00000E+00	5.03815E-02

MODE NO. 5

.00000E+00	-1.00952E+00	-1.03840E-01	.00000E+00	.00000E+00
-4.99923E-01	-9.70934E-02	.00000E+00	.00000E+00	7.75425E-02
-8.03535E-02	.00000E+00	.00000E+00	5.73384E-01	-5.33387E-02
.00000E+00	.00000E+00	8.58514E-01	-1.87125E-02	.00000E+00
.00000E+00	8.58514E-01	1.87125E-02	.00000E+00	.00000E+00
5.73384E-01	5.33387E-02	.00000E+00	.00000E+00	7.75426E-02
8.03535E-02	.00000E+00	.00000E+00	-4.99923E-01	9.70934E-02
.00000E+00	.00000E+00	-1.00952E+00	1.03840E-01	.00000E+00

MODE NO. 6

1.00952E+00	.00000E+00	.00000E+00	-1.03840E-01	4.99923E-01
.00000E+00	.00000E+00	-9.70934E-02	-7.75425E-02	.00000E+00
.00000E+00	-8.03535E-02	-5.73384E-01	.00000E+00	.00000E+00
-5.33387E-02	-8.58514E-01	.00000E+00	.00000E+00	-1.87125E-02
-8.58514E-01	.00000E+00	.00000E+00	1.87125E-02	-5.73384E-01
.00000E+00	.00000E+00	5.33387E-02	-7.75426E-02	.00000E+00
.00000E+00	8.03535E-02	4.99923E-01	.00000E+00	.00000E+00
9.70934E-02	1.00952E+00	.00000E+00	.00000E+00	1.03840E-01

* * * ROTOR GROUP NATURAL FREQUENCIES - WR * * *

RAD/SEC	HERTZ
2.94585E-03	4.68847E-04
2.94585E-03	4.68847E-04
3.64288E-03	5.79782E-04
3.64288E-03	5.79782E-04
3.09484E+03	4.92559E+02
3.09484E+03	4.92559E+02

* * * CASING GROUP NATURAL FREQUENCIES - WC * * *

RAD/SEC	HERTZ
2.53371E-03	4.03252E-04
2.53371E-03	4.03252E-04
3.13376E-03	4.98753E-04
3.13376E-03	4.98753E-04
7.67335E+03	1.22125E+03
7.67335E+03	1.22125E+03

* * * GYROSCOPIC MATRIX ELEMENTS * * *

ROW	COL	VALUE
3	4	-3.57800000E+00
4	3	3.57800000E+00
7	8	-7.15500000E+00
8	7	7.15500000E+00
11	12	-8.27000000E+01
12	11	8.27000000E+01
15	16	-7.15500000E+00
16	15	7.15500000E+00
19	20	-7.15500000E+00
20	19	7.15500000E+00
23	24	-1.60300000E+02
24	23	1.60300000E+02
27	28	-7.15500000E+00
28	27	7.15500000E+00
31	32	-7.15500000E+00
32	31	7.15500000E+00
35	36	-2.64400000E+02
36	35	2.64400000E+02
39	40	-3.57800000E+00
40	39	3.57800000E+00

* * * SUBCASE INPUT * * *

DAMPED CRITICAL SPEEDS AND STABILITY-10 LUMPED MASS ROTOR FLEX CASE
HPOTP LINEAR ADDITIONS TO MODEL/FLEXIBLE CASING

NUMBER OF GENERAL MATRIX ADDITIONS (STIFFNESS AND DAMPING)

PARTITION 1 = 0 PARTITION 2 = 0

PARTITION 3 = 0 PARTITION 4 = 4

NUMBER OF INTERGROUP ADDITIONS = 5

NUMBER OF FUNCTION GENERATORS = 10

MODE SHAPE PRINTING CODE = 1 NEXT CASE CODE = 4

RPM1 = 5.00000E+03 DRPM = .00000E+00 IP = 1

* * * CASING GROUP MODAL DAMPING FACTORS * * *

1.00000E-02
1.00000E-02
1.00000E-02
1.00000E-02
1.00000E-02
1.00000E-02

* * * FUNCTION GENERATOR DATA * * *

FUNCTION NO. 1 UNITY$

RPM	VALUE
1.0000E+03	1.0000E+00
9.0000E+04	1.0000E+00

FUNCTION NO. 2 PBP RWR DIRECT STIFFNESS$

RPM	VALUE
1.0000E+03	.0000E+00
1.0000E+04	1.0000E+05
2.0000E+04	4.2000E+05
2.5000E+04	6.5000E+05
3.0000E+04	9.3000E+05
3.5000E+04	1.2400E+06
4.0000E+04	1.6000E+06
4.5000E+04	2.1000E+06

FUNCTION NO. 3 PBP RWR C-CSTIFFNESS$

RPM	VALUE
1.0000E+03	.0000E+00
1.0000E+04	1.4900E+04
2.0000E+04	5.8000E+04
2.5000E+04	8.6000E+04
3.0000E+04	1.2000E+05
3.5000E+04	1.5900E+05
4.0000E+04	2.0400E+05
4.5000E+04	2.5800E+05

FUNCTION NO. 4 PBP RWR DIRECT DAMPING$

RPM	VALUE
1.0000E+03	.0000E+00
1.0000E+04	6.3380E+01
3.0000E+04	2.1000E+02
4.5000E+04	3.2100E+02

FUNCTION NO. 5 M IMP DIRECT STIFFNESS$

RPM	VALUE
1.0000E+03	8.8900E+01
5.0000E+03	2.2222E+03
1.0000E+04	8.8889E+03
1.5000E+04	2.0000E+04
2.0000E+04	3.5556E+04
2.5000E+04	5.5556E+04
3.0000E+04	8.0000E+04
3.5000E+04	1.0889E+05
4.0000E+04	1.4222E+05
4.5000E+04	1.8000E+05

FUNCTION NO. 6 M IMP C-C STIFFNESS$

RPM	VALUE
1.0000E+03	4.0000E+01
5.0000E+03	1.0000E+03
1.0000E+04	4.0000E+03
1.5000E+04	9.0000E+03
2.0000E+04	1.6000E+04
2.5000E+04	2.5000E+04
3.0000E+04	3.6000E+04
3.5000E+04	4.9000E+04
4.0000E+04	6.4000E+04
4.5000E+04	8.1000E+04

FUNCTION NO. 7 M IMP DIRECT DAMPING$

RPM	VALUE
1.0000E+03	1.0000E-01
5.0000E+03	5.0000E-01
1.0000E+04	1.0000E+00
1.5000E+04	1.5000E+00
2.0000E+04	2.0000E+00
2.5000E+04	2.5000E+00
3.0000E+04	3.0000E+00
3.5000E+04	3.5000E+00
4.0000E+04	4.0000E+00
4.5000E+04	4.5000E+00

FUNCTION NO. 8 TURBINE DIRECT STIFFNESS$

RPM	VALUE
1.0000E+03	.0000E+00
5.0000E+03	2.1800E+03
1.0000E+04	8.7100E+03
1.5000E+04	1.9600E+04
2.0000E+04	3.4850E+04
2.5000E+04	4.7500E+04
3.0000E+04	6.8300E+04
3.5000E+04	9.5500E+04
4.0000E+04	1.2650E+05
4.5000E+04	1.6100E+05

FUNCTION NO. 9 TURBINE C-C STIFFNESS$

RPM	VALUE
1.0000E+03	.0000E+00
5.0000E+03	7.1000E+02
1.0000E+04	1.9300E+03
1.5000E+04	3.9700E+03
2.0000E+04	7.8800E+03
2.5000E+04	1.3250E+04
3.0000E+04	2.1100E+04
3.5000E+04	3.0950E+04
4.0000E+04	4.6000E+04
4.5000E+04	6.4300E+04

FUNCTION NO. 10 TURBINE DIRECT DAMPINGS$

RPM	VALUE
1.0000E+03	.0000E+00
5.0000E+03	9.0000E-01
1.0000E+04	2.3000E+00
1.5000E+04	4.2000E+00
2.0000E+04	6.1000E+00
2.5000E+04	8.8000E+00
3.0000E+04	1.2800E+01
3.5000E+04	1.7300E+01
4.0000E+04	2.3000E+01
4.5000E+04	2.9600E+01

* * * * * * * * MATRIX ADDITIONS * * * *

* * * INTERGROUP ADDITIONS * * *

UPPER LEFT PARTITION

NO.	ROW	COL.	STIFFNESS	FUNC.	DAMPING	FUNC.
1	1	1	5.00000E+05	1	2.50000E+00	1
2	2	2	5.00000E+05	1	2.50000E+00	1
3	1	2	.00000E+00	1	.00000E+00	1
4	2	1	.00000E+00	1	.00000E+00	1
5	37	37	5.00000E+05	1	2.50000E+00	1
6	38	38	5.00000E+05	1	2.50000E+00	1
7	37	38	.00000E+00	1	.00000E+00	1
8	38	37	.00000E+00	1	.00000E+00	1
9	9	9	1.00000E+00	2	1.00000E+00	4
10	10	10	1.00000E+00	2	1.00000E+00	4
11	9	10	1.00000E+00	3	.00000E+00	1
12	10	9	-1.00000E+00	3	.00000E+00	1
13	21	21	-1.00000E+00	5	1.00000E+00	7
14	22	22	-1.00000E+00	5	1.00000E+00	7
15	21	22	1.00000E+00	6	.00000E+00	1
16	22	21	-1.00000E+00	6	.00000E+00	1
17	33	33	1.00000E+00	8	1.00000E+00	10
18	34	34	1.00000E+00	8	1.00000E+00	10
19	33	34	1.00000E+00	9	.00000E+00	1
20	34	33	-1.00000E+00	9	.00000E+00	1

UPPER RIGHT PARTITION

NO.	ROW	COL.	STIFFNESS	FUNC.	DAMPING	FUNC.
1	1	1	-5.00000E+05	1	-2.50000E+00	1
2	2	2	-5.00000E+05	1	-2.50000E+00	1
3	1	2	.00000E+00	1	.00000E+00	1
4	2	1	.00000E+00	1	.00000E+00	1
5	37	37	-5.00000E+05	1	-2.50000E+00	1
6	38	38	-5.00000E+05	1	-2.50000E+00	1
7	37	38	.00000E+00	1	.00000E+00	1
8	38	37	.00000E+00	1	.00000E+00	1
9	9	9	-1.00000E+00	2	-1.00000E+00	4
10	10	10	-1.00000E+00	2	-1.00000E+00	4
11	9	10	-1.00000E+00	3	.00000E+00	1
12	10	9	1.00000E+00	3	.00000E+00	1
13	21	21	1.00000E+00	5	-1.00000E+00	7
14	22	22	1.00000E+00	5	-1.00000E+00	7
15	21	22	-1.00000E+00	6	.00000E+00	1
16	22	21	1.00000E+00	6	.00000E+00	1
17	33	33	-1.00000E+00	8	-1.00000E+00	10
18	34	34	-1.00000E+00	8	-1.00000E+00	10
19	33	34	-1.00000E+00	9	.00000E+00	1
20	34	33	1.00000E+00	9	.00000E+00	1

LOWER LEFT PARTITION

NO.	ROW	COL.	STIFFNESS	FUNC.	DAMPING	FUNC.
1	1	1	-5.00000E+05	1	-2.50000E+00	1
2	2	2	-5.00000E+05	1	-2.50000E+00	1
3	1	2	.00000E+00	1	.00000E+00	1
4	2	1	.00000E+00	1	.00000E+00	1
5	37	37	-5.00000E+05	1	-2.50000E+00	1
6	38	38	-5.00000E+05	1	-2.50000E+00	1
7	37	38	.00000E+00	1	.00000E+00	1
8	38	37	.00000E+00	1	.00000E+00	1
9	9	9	-1.00000E+00	2	-1.00000E+00	4
10	10	10	-1.00000E+00	2	-1.00000E+00	4
11	9	10	-1.00000E+00	3	.00000E+00	1
12	10	9	1.00000E+00	3	.00000E+00	1
13	21	21	1.00000E+00	5	-1.00000E+00	7
14	22	22	1.00000E+00	5	-1.00000E+00	7
15	21	22	-1.00000E+00	6	.00000E+00	1
16	22	21	1.00000E+00	6	.00000E+00	1
17	33	33	-1.00000E+00	8	-1.00000E+00	10
18	34	34	-1.00000E+00	8	-1.00000E+00	10
19	33	34	-1.00000E+00	9	.00000E+00	1
20	34	33	1.00000E+00	9	.00000E+00	1

LOWER RIGHT PARTITION

NO.	ROW	COL.	STIFFNESS	FUNC.	DAMPING	FUNC.
1	1	1	5.00000E+05	1	2.50000E+00	1
2	2	2	5.00000E+05	1	2.50000E+00	1
3	1	2	.00000E+00	1	.00000E+00	1
4	2	1	.00000E+00	1	.00000E+00	1
5	37	37	5.00000E+05	1	2.50000E+00	1
6	38	38	5.00000E+05	1	2.50000E+00	1
7	37	38	.00000E+00	1	.00000E+00	1
8	38	37	.00000E+00	1	.00000E+00	1
9	9	9	1.00000E+00	2	1.00000E+00	4
10	10	10	1.00000E+00	2	1.00000E+00	4
11	9	10	1.00000E+00	3	.00000E+00	1
12	10	9	-1.00000E+00	3	.00000E+00	1
13	21	21	-1.00000E+00	5	1.00000E+00	7
14	22	22	-1.00000E+00	5	1.00000E+00	7
15	21	22	1.00000E+00	6	.00000E+00	1
16	22	21	-1.00000E+00	6	.00000E+00	1
17	33	33	1.00000E+00	8	1.00000E+00	10
18	34	34	1.00000E+00	8	1.00000E+00	10
19	33	34	1.00000E+00	9	.00000E+00	1
20	34	33	-1.00000E+00	9	.00000E+00	1

* * * GENERAL ADDITIONS * * *

LOWER RIGHT PARTITION

NO.	ROW	COL.	STIFFNESS	FUNC.	DAMPING	FUNC.
1	5	5	1.00000E+06	1	.00000E+00	1
2	6	6	1.00000E+07	1	.00000E+00	1
3	33	33	1.00000E+06	1	.00000E+00	1
4	34	34	1.00000E+07	1	.00000E+00	1

ROTOR DISPLACEMENTS WILL BE COMPUTED AT THE FOLLOWING DEGREES OF FREEDOM

```
 1   2   5   6   9  10  13  14  17  18  21  22
25  26  29  30  33  34  37  38
```

CASING DISPLACEMENTS WILL BE COMPUTED AT THE FOLLOWING DEGREES OF FREEDOM

```
 1   2   5   6   9  10  13  14  17  18  21  22
25  26  29  30  33  34  37  38
```

* * PRINT AND PLOT CONTROL OPTIONS * *

```
FHIGH =  5.00000E+04 CPM      FLOW =  1.00000E+04 CPM

XB    =  5.00000E+03          XR     =  6.00000E+04
YB    =  5.00000E+03          YT     =  6.00000E+04
DX    =  5.00000E+03          DY     =  5.00000E+03

IPLTF =  0                    IPRT2 =  0

XB1    =  5.00000E+03          XR1    =  6.00000E+04
YB1    = -2.00000E+02          YT1    =  2.00000E+02
DX1    =  5.00000E+03          DY1    =  5.00000E+01

STABILITY PLOT CHARACTERS        1      2      3      4

IPRT3 =  0

MODE SHAPES PLOTTED FOR  2 SPEED CASES AT FOLLOWING SPEEDS

         10000.     30000.

MODE SHAPE PLOTTING DATA

STATIONS= 10    IFLG= 3    IFLG1= 3    THETA= 210.0    SCALE= .050

STATION LOCATIONS

           .000      3.000      6.000      9.000     12.000
         15.000     18.000     21.000     24.000     27.000
```

APPENDIX D

PROGRAM RSTAB USER'S GUIDE

SUMMARY

The development of asn IBM PC version of the CDC program DAMCISS is complete. This work was done to support NAS8-34964, titled "Effects of Case Flexibility on Bearing Loads and Rotor Stability Study," and was also designed to be of general use in the analysis of current rotordynamic models and in future studies. Considerable modification and revision to the program structure and subroutines was necessary to accomplish this. The code has been tested and verified using three independent models and has demonstrated very good agreement with the CDC equivalent program.

This Appendix discusses the analytical methods used, which include substructure modeling, general stiffness and damping modifications and complex eigenvalue solution. Discussion of program implementation on the IBM PC computer includes memory size, execution speed and disk storage considerations. A main memory capacity of 256K bytes, an output buffer spooling program and temporary disk storage space are essential to efficient program use for large models (typical of current SSME turbopump analyses). However, small models used for generic study purposes may be run efficiently under more restricted configurations. Redimensioning of the program can be done by the experienced user, thus enabling adaptation of the code to various model sizes and computer configurations. A guide has been provided for this, which relies on the DAMCISS User's Guide for definition of the variables involved.

Program execution, using the batch file RUNRSTAB, is described. Some familiarity with the IBM operating system will be needed to vary from the rigid procedure provided, but it should be useful in getting anyone started. An example run session has been provided to assist in this.

GENERAL PROGRAM DESCRIPTION

RSTAB allows independently formulated component (rotor and casing) models to be combined using normalized modes into a coupled system. Interaction forces between the substructures, as well as general stiffness and damping modifications internal to the substructures, are specified in the physical coordinate system and transformed by the program into the normalized coordinate system. The coupled system is then solved for its complex eigenvalues and eigenvectors, which yield the required information regarding the critical speeds and stability margins of the system. Via backtransformation to the physical coordinate system, the resulting mode shapes and the direction of rotor precession are computed. By permitting the specification of RPM-dependency of each system addition and modification, the behavior of such turbopump interaction elements as bearings, seals and turbines, can be accurately modeled. The eignevalue solution process is then repeated while parametrically varying the pump speed.

A root-locus plot is produced as the program executes. This plot combines the information contained in the critical speed and stability plots and also provides visual feedback on the program's execution status. The critical damping ratio of each mode can be graphically determined from the root-locus plot. Lines of equal damping are straight lines extending radially from the plot origin. Lines for 1, 2.5, 5 and 10% of critical damping are indicated on each plot.

Microsoft FORTRAN Version 3.13 was used to develop the programs. The philosophy of program organization has been to enhance efficiency in the desktop computer environment and to provide clear subdivisions for chaining the programs to operate under main memory restrictions (the computer used for development was an IBM PC with 256K bytes of main memory).

The code has been divided into three executable programs, which are executed sequentially using the batch procedure RUNRSTAB. The first of these three (PRERSTAB) entirely pre-processes the input files, including all subcase iterations. It creates a binary run data file, which is then used by the main program (RSTAB). The main program, which solves for the coupled system eigenvalues and mode shapes as operating speed is varied, produces a root-locus plot (upper

half-plane) for each subcase run. This plot is the only graphics produced by the main program. All other results information is written to binary files for use by the post-processor program (PSTRSTAB). This last program produced all remaining plotted output, including function generator curves, critical speed and stability plots and complex mode shapes. Restart capability is afforded by the various binary files created by each program, which can be saved for this purpose.

All plotted output is directed to a dot-matrix printer, which must be present for execution of RSTAB (for the root-locus plot) and PSTRSTAB (if any other plots are requested). The output listing, however, may be printed as the programs execute (immediate printing), or may be directed to disk storage.

The amount of main memory required to execute the programs is determined by the size of the main program (RSTAB), which is the largest of the three. The size of this program as currently dimensioned is 198,544 bytes. This includes plotting and CRT control functions which are not mandatory for successful execution of the three programs if root-locus plots are not desired. A non-plotting version of the main program (RSTABNP) has also been provided, and requires only 177,120 bytes. Altering the dimensioning of variable arrays to accommodate problems of different sizes would affect the program size. Table C.1 details the dimensioning of Version 1.0 of RSTAB.

RESTAB REDIMENSIONING

Array redimensioning is the most likely modification to be necessary on a routine basis. Three source files will require modification for redimensioning. These are: PRERSTAB.FOR, RSTAB.FOR, and PSTRSTAB.FOR. Note that these need not be dimensioned identically. The only requirement is that each be dimensioned to at least the sizes required for the problem to be analyzed (such that the user's input does not exceed the capacity of any one program). The user must be aware that any other software modification may adversely affect the program operation and validity. A guide is provided for the experienced user to redimension the program.

After recompiling the modified source codes, the executable files can be re-linked by following the linking guides also provided. The object modules TKNULL, EPNULL and NULE6 are PLOT88 and FORTRAN "null" object modules (to eliminate unnecessary code) and should be used when linking the main program (RSTAB).

PROGRAM USAGE

The IBM PC program XTALK should be used to transmit input data files from the CDC computer. With XTALK in "Capture" mode, any formatted files can be transmitted by using the CDC NOS command "COPY, file" (where "file" is the local name of the CDC formatted file). For transmitting binary mode shape files created by the finite-element analysis program V9568, the following procedure is provided:

```
GET, V956IBM/UN = YQA314
V956IBM
```

This procedure will invoke a translator program, which will read the binary file and transmit it formatted for use by program PRERSTAB. All leading and trailing lines in input files (including CDC JCL) will be ignored by RSTAB; therefore, no special editing of the imput files will be necessary.

An IBM batch file, RUNRSTAB, has been provided to facilitate the execution of the three programs. The user should execute this batch file from a directory with enough space for program output and temporary files (this requirement will vary considerably with problem size and plot requests). RUNRSTAB will pause with messages to direct the interchange of the diskettes containing the executable files. Provided with the executable files is a program usage log (SAV.LOG) for productivity accounting. This file is mandatory for program execution. An example run session is included.

Execution speed is greatly enhanced if the output listing is directed to disk storage, or if a spooler program is used to establish an output buffer of at least 30 Kbytes. For this reason it is recommended to execute under the DOS 1.1 operating system using Superspool, and have at least 256 Kbytes of main memory. With this amount of memory, 26 Kbytes of output buffer can be allocated, leaving enough memory for execution of RSTAB.

TABLE C.1. RESTAB VERSION 1.0 DIMENSIONING

ROTOR MODES	16
ROTOR DEGREES OF FREEDOM	115
CASING MODES	14
CASING DEGREES OF FREEDOM	50
FUNCTION GENERATORS	24
POINTS PER FUNCTION GENERATOR	20
RPM INCREMENTS	50
INTRAGROUP STRUCTURAL ADDITIONS	20
INTERGROUP STRUCTURAL ADDITIONS	20
ROTOR DEGREES-OF-FREEDOM FOR PRINTED OUTPUT	20
CASING DEGREES-OF-FREEDOM FOR PRINTED OUTPUT	20
ROTOR/CASING STATIONS FOR PLOTTED OUTPUT	10
RPM STEPS FOR MODE SHAPE PRINTING/PLOTTING	25

1. PROGRAM RSTAB RE-DIMENSIONING GUIDE

In order to run problems requiring larger array dimensions than provided in Version 1.0 of RSTAB, or to run smaller problems using less main memory, it will be necessary to re-dimension the program. Only the three principal source files PRERSTAB.FOR, RSTAB.FOR, (or RSTABNP.FOR for non-plotting version) and PSTRSTAB.FOR will need any modification for this. It is not necessary to dimension these identically, provided that each is dimensioned large enough to accommodate the problem at hand.

For definition of variables not defined in this guide, refer to the DAMCISS User's Guide . These variables are identified by *italics* type.

I. Program File PRERSTAB.FOR

ARRAYS

TYPE	NAME	DIMENSION(S)
INTEGER*2	JFUN	(LSJ,2)
INTEGER*2	~~NTP~~ NPT	(*NFGEN*)
INTEGER*2	NRC	(LSA,2,4)
INTEGER*2	IROT	(*NIROT*)
INTEGER*2	ICASE	(*NCASE*)
REAL*4	AC	(*NCDOF*, *NCMOD*)
REAL*4	AR	(*NRDOF*, *NRMOD*)
REAL*4	WC, ZETAC	(*NCMOD*)
REAL*4	WR	(*NRMOD*)
REAL*4	DMP,S	(LSA,4)
REAL*4	SPEED,FG	(LPT, *NFGEN*)
REAL*4	G	(*NRDOF*, *NRDOF*)
REAL*4	ZRPM	(*KRPM*)
REAL*4	X	(*NSTAT*)
REAL*8	GBAR	(*NRMOD*, *NRMOD*)
REAL*8	WK	(*NRDOF*)

VARIABLES IN DATA STATEMENTS (ALL INTEGER*2)

LCMOD = *NCMOD*

LRMOD = *NRMOD*

LRDOF = *NRDOF*

LCDOF = *NCDOF*

LSA = 4**NSL* + MAXØ (*NSA1*, *NSA2*, *NSA3*, *NSA4*)

LSJ = 4**NSL* + *NSA1* + *NSA2* + *NSA3* + *NSA4*

LPT = Maximum number of points in any function generator

LINKING

OBJECT MODULES: PRERSTAB
FORTØ1
FORTØ2

LIBRARIES: GRAFMS3
FORTRAN

II. Program File RSTAB.FOR

ARRAYS

TYPE	NAME	DIMENSIONS*
INTEGER*2	JFUN	SAME AS IN PRERSTAB.FOR
INTEGER*2	NPT	SAME AS IN PRERSTAB.FOR
INTEGER*2	NRC	SAME AS IN PRERSTAB.FOR
INTEGER*2	IROT	SAME AS IN PRERSTAB.FOR
INTEGER*2	ICASE	SAME AS IN PRERSTAB.FOR
REAL*4	AC	SAME AS IN PRERSTAB.FOR
REAL*4	AR	SAME AS IN PRERSTAB.FOR
REAL*4	WC, ZETAC	SAME AS IN PRERSTAB.FOR
REAL*4	WR	SAME AS IN PRERSTAB.FOR
REAL*4	DMP,S	SAME AS IN PRERSTAB.FOR
REAL*4	SPEED,FG	SAME AS IN PRERSTAB.FOR
REAL*4	G	SAME AS IN PRERSTAB.FOR
REAL*4	ZRPM	SAME AS IN PRERSTAB.FOR
REAL*4	X	SAME AS IN PRERSTAB.FOR
REAL*4	W	(2,LDYN)
REAL*4	FUNC	(*NFGEN* + 1)
REAL*8	GBAR	SAME AS IN PRERSTAB.FOR
REAL*8	WK	(LWK)
REAL*8	A	(LDYN, LDYN)
REAL*8	Z	(2, LDYN, LDYN)

Where: LDYN = 2*(*NRMOD* + *NCMOD*) AND
LWK = MAXØ(2*LDYN, *NRDOF*, *NCDOF*)

VARIABLES IN DATA STATEMENTS

Same as for PRERSTAB.FOR

LINKING

OBJECT MODULES:	RSTAB (or RSTABNP for non-plotting version) FORTØ3 (or FORTØ3NP for non-plotting version) EIGRF TKNULL EPNULL NULE6
LIBRARIES:	GRAFMS3 (required when linking with RSTAB) PLOT88 (required when linking with RSTAB) FORTRAN

III. Program File PSTRSTAB.FOR

ARRAYS

TYPE	NAME	DIMENSION(S)
INTEGER*2	IROT	SAME AS IN PRERSTAB.FOR
INTEGER*2	ICASE	SAME AS IN PRERSTAB.FOR
REAL*4	AC	SAME AS IN PRERSTAB.FOR
REAL*4	AR	SAME AS IN PRERSTAB.FOR
REAL*4	X	SAME AS IN PRERSTAB.FOR
REAL*4	FG,SP	(LPT + 2)
REAL*4	VECT	(2, *NIROT* + *NCASE*)
REAL*4	W	(2, *NRMOD* + *NCMOD*, *NI* + 1)
REAL*4	Z	(2, *NRMOD* + *NCMOD*)

Where: LPT = Maximum number of points in any function generator.

VARIABLES IN DATA STATEMENTS (ALL INTEGER*2)

LRDOF = *NRDOF*

LCDOR = *NCDOF*

LMOD = *NRMOD* + *NCMOD*

LINKING

OBJECT MODULES:	PSTRSTAB SPDPLT SHHPLT
LIBRARIES:	GRAFMS3 PLOT88 FORTRAN

2. EXAMPLE RSTAB RUN SESSION USING BATCH FILE RUNRSTAB

Use of the RUNRSTAB batch file is a nine-step process. User action is only required in the first five steps.

STEP 1

Copy the batch file into the run directory (one with enough space to execute the programs).

ENTER:

```
COPY F:RUNRSTAB.BAT
        1 File(s) copied
```

STEP 2

Run the batch file. The parameter required ("F" in this example) is the drive letter for the removable disks containing the executable files.

ENTER:

```
RUNRSTAB F
```

STEP 3

Respond to the batch file prompts directing the interchange of disks containing executable files.

```
D>Pause - INSERT RSTAB/PSTRSTAB DISK INTO DRIVE F
Strike a key when ready . . .
D>COPY F:RSTAB.EXE
        1 File(s) copied
D>COPY F:PSTRSTAB.EXE
        1 File(s) copied
D>Pause - INSERT PRERSTAB DISK INTO DRIVE F
Strike a key when ready . . .
D>F:PRERSTAB
```

STEP 4

PRERSTAB displays a program banner, then prompts for three inputs. The output listing file name many be "PRN" for immediate printing of output data.

```
*****************************************************************************
*****                                                                   ****
*****                  P R O G R A M   R S T A B                        ****
*****                                                                   ****
*****                  IBM PC VERSION 1.0                               ****
*****                                                                   ****
*****                  APRIL 1984                                       ****
*****                                                                   ****
*****************************************************************************
DATE:  7/26/1984                                     BEGIN TIME: 16:58:4




ENTER OUTPUT LISTING FILE NAME:  LIST.DAT

ENTER FILE NAME FOR INPUT DATA:  F:HP26CSS.CDC

ENTER DRIVE CONTAINING THE PROGRAM USAGE LOG:  F
```

STEP 5

PRERSTAB then displays the modal section input data and prompts for the file(s) containing the input normal modes. This will be repeated for each modal input section in the user's input file.

```
   * * * * M O D A L   I N P U T   S E C T I O N   1 * * * *

SSME HPOTP 26000 RPM REDESIGN APR. 84

   NO. OF ROTOR DOF  = 113     NO. OF ROTOR MODES  =  13
   NO. OF CASING DOF =  39     NO. OF CASING MODES =  14
   ROTOR TAPE CODE   =   1     CASING TAPE CODE    =   0
   NO. OF GYROSCOPIC ADDITIONS =  64
   ACCELERATION OF GRAVITY     =  386.40

ENTER THE ROTOR MODE SHAPE FILE NAME:  F:HP26MDO.CDC
```

STEP 6

PRERSTAB then displays the subcase input data. This will be repeated for each subcase input section in the user's input file.

```
* * * *   S U B C A S E   I N P U T   * * * *

            MODAL INPUT 1     SUBCASE 1

SSME HPOTP 26000 RPM REDESIGN APR. 84

S R PREB SEALS, R T SEAL, OV IN 0.5

NUMBER OF GENERAL MATRIX ADDITIONS:

     PARTITION 1 =   2          PARTITION 2 =   2
     PARTITION 3 =   2          PARTITION 4 =   2

NUMBER OF INTERGROUP ADDI1IONS =  14

NUMBER OF FUNCTION GENERATORS  =  24

MODE SHAPE PRINTING CODE =   1

NEXT CASE CODE =   4

RPM1  =  1.00000E+04        DRPM  =  1.00000E+03        IP  =   0
```

All user inputs and input files have been read in as of this point, and the program will run until completion. A root-locus plot will be displayed for each subcase, indicating the status of the run.

STEP 7

Upon completion of PRERSTAB, the batch file next executes RSTAB.

```
D>RSTAB
```

STEP 8

Upon completion of RSTAB, the batch file will delete the RSTAB program and RUNDATA files from the run directory (to conserve space). Then the post-processor program will be executed.

```
D>DEL RSTAB.EXE

D>DEL RUNDATA.BIN

D>PSTRSTAB
```

```
* * * *  S U B C A S E   P L O T T I N G  * * * *

            MODAL INPUT 1     SUBCASE 1

SSME HPOTP 26000 RPM REDESIGN APR. 84

S R PREB SEALS, R T SEAL, OV IN 0.5

NO. OF ROTOR DOF  = 113     NO. OF ROTOR MODES  =  13

NO. OF CASING DOF =  39     NO. OF CASING MODES =  14

RPM1  =  1.00000E+04        DRPM  =  1.00000E+03        IP  =  0
```

All plotable output, for each subcase, will be output to the dot-matrix printer.

STEP 9

Upon completion of PSTRSTAB, a completion banner will be output. Then PSTRSTAB will execute the wrap-up batch file (FIN), which will copy the output listings together (if output was directed to disk storage).

```
***************************************************************************
*****                                                                  ****
*****                   EXECUTION COMPLETED AT 17:31:29                ****
*****                                                                  ****
***************************************************************************

D>FIN

D>COPY LIST.DAT       +LIST2+LIST3
        1 File(s) copied
D>DEL LIST2

D>DEL LIST3

D>
D>
```

3. DAMCISS USER'S GUIDE

1		
13		
25	TITLE(I,1)	7A10,1A2
37		
49		
61		
IDENTIFICATION 73	100010	80

PROBLEM TITLE CARD

1 CARD REQUIRED

MAY OCCUPY COLUMNS 1 – 72

NOTE: THE FIRST 60 CHARACTERS OF THIS TITLE WILL APPEAR ON ALL PLOTS CREATED BY THIS PROGRAM

MODEL SIZE CARD

1	NRDØF	NRMØD	7I6
13	NCDØF	NCMØD	
25	NGYRØ	NRTAP	
37	NCTAP	GRAV	E12.6
49		MLIST	I6
61			
	IDENTIFICATION 73	100020	80

NRDØF = NO. OF ROTOR DEGREES OF FREEDOM ($\leq$ 125)

NRMØD = NO. OF ROTOR MODE SHAPES ($\leq$ 20)

NCDØF = NO. OF CASING DEGREES OF FREEDOM ($\leq$ 125)

NCMØD = NO. OF CASING MODE SHAPES ($\leq$ 20)

NGYRØ = NO. OF GYROSCOPIC MATRIX ADDITIONS

NRTAP = FLAG FOR READING ROTOR MODES AND FREQUENCIES

NRTAP = 0 ROTOR MODES AND FREQUENCIES WILL BE READ FROM CARDS

NRTAP = 1 ROTOR MODES AND FREQUENCIES WILL BE READ FROM TAPE4, CREATED IN A PREVIOUS V9568 RUN. CARDS 5XXX AND 601XX NOT REQUIRED.

NRTAP = 2 ROTOR MODES AND FREQUENCIES WILL BE READ FROM TAPE4, CREATED IN A PREVIOUS STARDYNE RUN. CARDS 5XXX AND 601XX NOT REQUIRED.

NRTAP = 3 ROTOR MODES ONLY WILL BE READ FROM TAPE1, CARD 5XXX FORMAT. CARDS 5XXX NOT REQUIRED.

NOTE: IF NRTAP = 1, NRMOD MUST EQUAL THE NUMBER OF MODES REQUESTED IN THE V9568 RUN.

MODEL SIZE CARD (CONTINUED)

NCTAP = FLAG FOR READING CASING MODES AND FREQUENCIES

NCTAP = 0 CASING MODES AND FREQUENCIES WILL BE READ FROM CARDS.

NCTAP = 1 CASING MODES AND FREQUENCIES WILL BE READ FROM TAPE 3, CREATED BY A PREVIOUS V9568 RUN. CARDS 55XXX AND 602XX NOT REQUIRED.

NCTAP = 2 CASING MODES AND FREQUENCIES WILL BE READ FROM TAPE 3, CREATED BY A PREVIOUS STARDYNE RUN. CARDS 55XXX AND 602XX NOT REQUIRED.

NCTAP = 3 CASING MODES ONLY WILL BE READ FROM TAPE 1, CARD 55XXX FORMAT. CARDS 55XXX NOT REQUIRED.

NOTE: IF NCTAP = 1, NCMOD MUST EQUAL THE NUMBER OF MODES REQUESTED IN THE V9568 RUN.

GRAV = ACCELERATION OF GRAVITY
NORMALLY GRAV = 386.4

MLIST = FLAG FOR PRINTING OF ROTOR/CASING MODE SHAPES FROM INPUT FILES

MLIST = 0 PRINT MODE SHAPES (DEFAULT)
MLIST = 1 DO NOT PRINT MODE SHAPES

ROTOR MODE SHAPES

1	AR(I, J)	
13	⋮	
25	⋮	6E12.8
37	⋮	
49	⋮	
61	AR(NRDOF, NRMOD)	
	IDENTIFICATION	73 5XXX 80

REPEAT THIS CARD UNTIL NRDOF ENTRIES ARE MADE FOR EACH OF NRMOD ROTOR MODES.

REQUIRED ONLY IF NRTAP = 0

NOTE: THESE MODES MUST BE NORMALIZED SUCH THAT

$$[\Phi]^T [M_R] [\Phi] = [I]$$

CASING MODE SHAPES

1	AC(I, J)	
13	⋮	
25	⋮	6E12.8
37	⋮	
49	⋮	
61	AC(NCDOF, NCMOD)	
	IDENTIFICATION	73 55XXX 80

REPEAT THIS CARD UNTIL NCDOF ENTRIES ARE MADE FOR EACH OF NCMOD CASING MODES.

REQUIRED ONLY IF NCTAP = 0

NOTE: THESE MODES MUST BE NORMALIZED SUCH THAT

$$[\Phi]^T [M_C] [\Phi] = [I]$$

ROTOR NATURAL FREQUENCIES

1	WR(1)	
13	⋮	
25	⋮	6E12.8
37	⋮	
49	⋮	
61	WR(NRMOD)	
	IDENTIFICATION 73	601XX 80

REPEAT THIS CARD UNTIL NRMOD ROTOR FREQUENCIES ARE ENTERED. UNITS OF WR ARE RADIANS/SEC.

REQUIRED IF NRTAP = 0 OR 3.

CASING NATURAL FREQUENCIES

1	WC(1)	
13	⋮	
25	⋮	6E12.8
37	⋮	
49	⋮	
61	WC(NCMOD)	
	IDENTIFICATION 73	602XX 80

REPEAT THIS CARD UNTIL NCMOD CASING FREQUENCIES ARE ENTERED. UNITS OF WC ARE RADIANS/SEC.

REQUIRED IF NCTAP = 0 OR 3.

FORM R 152-R-2 REV. 10-73

PREPARED BY:	Rocketdyne Division Rockwell International	PAGE NO. 6
CHECKED BY:		REPORT NO.
DATE:		

GYROSCOPIC MATRIX ADDITIONS

Col.		
1	IR IJ	2I6
13	GYRØ(IR, IJ)	E12.8
25		
37		
49		
61		
IDENTIFICATION 73	6.03	80

IR = ROTOR GROUP ROW NUMBER

IJ = ROTOR GROUP COLUMN NUMBER

GYRO(IR, IJ) = WEIGHT POLAR MOMENT OF INERTIA TO BE ADDED AT THESE DOF'S

NGYRO CARDS REQUIRED
(NGYRO MUST BE ≥ 1)

NOTE: GYROSCOPIC ADDITIONS ARE NORMALLY ENTERED IN PAIRS, WITH A (+) MOMENT FOR IR > IJ AND A (−) MOMENT FOR IJ > IR

FORM R 152-R-2 REV. 10-73

1		
13		
25	TITLE(I,2)	7A10,1A2
37		
49		
61		
	IDENTIFICATION 73	60400020 80

SUPPLEMENTARY TITLE

1 CARD REQUIRED

NOTE: THE FIRST 60 CHARACTERS OF THIS TITLE WILL APPEAR ON ALL PLOTS CREATED BY THIS PROGRAM

CONNECTING ELEMENT DESCRIPTION CARD

1	NSA1	NSA2	
13	NSA3	NSA4	
25	NSL	IPRT	9I6
37	KCRT	NXTC	
49	NFGEN		
61			
	IDENTIFICATION 73	605	80

1 CARD REQUIRED

NSA1 = NO. OF UPPER LEFT PARTITION GENERAL STIFFNESS AND DAMPING ADDITIONS

NSA2 = NO. OF UPPER RIGHT PARTITION GENERAL STIFFNESS AND DAMPING ADDITIONS

NSA3 = NO. OF LOWER LEFT PARTITION GENERAL STIFFNESS AND DAMPING ADDITIONS

NSA4 = NO. OF LOWER RIGHT PARTITION GENERAL STIFFNESS AND DAMPING ADDITIONS

NSL = NO. OF STATIONS WHERE INTERGROUP ADDITIONS SUCH AS BEARINGS AND SEALS ARE TO BE ADDED

IPRT = CONTROL FLAG FOR PRINTING OF COUPLED SYSTEM MODE SHAPES

IPRT = 0 PRINT MODE SHAPES
IPRT = 1 NO PRINTING

CONNECTING ELEMENT DESCRIPTION CARD
(CONTINUED)

KCRT = CONTROL FLAG FOR STABILITY PLOTS

KCRT = 0 PLOT STABILITY CURVE OF 4 MODES PER FRAME

KCRT = 1 PLOT STABILITY CURVE OF ALL MODES ON 1 FRAME

NXTC = CONTROL FLAG FOR MULTIPLE CASE RUNS

NXTC = 1 UPON COMPLETION OF THIS CASE, RETURN TO READ CARD 100010 AND CONTINUE.

NXTC = 2 RETURN TO READ CARD 60400020 AND CONTINUE.

NXTC = 3 RETURN TO READ CARD 605XXXXX AND CONTINUE.

NXTC = 4 TERMINATE AFTER THIS CASE.

NFGEN = NO. OF FUNCTION GENERATORS FOR SPEED SCALING OF STIFFNESS AND DAMPING ADDITIONS
($NFGEN \leq 25$)

SPEED DESCRIPTION CARD

1	RPM1		2E12.8
13	DRPM		
25		NI	I12
37			
49			
61			
	IDENTIFICATION 73	6060	80

1 CARD REQUIRED

RPM1 = LOWEST ROTOR SPIN SPEED AT WHICH SYSTEM NATURAL FREQ'S SHOULD BE CALCULATED AND PRINTED

NOTE: PRINTING OF THE MODE SHAPES IS CONTROLLED BY IPRT ON CARD 605XX

DRPM = ROTOR SPIN SPEED INCREMENT TO NEXT SPEED FOR CALCULATING NATURAL FREQUENCIES

NI = TOTAL NUMBER OF INCREMENTS $(1 \leq NI \leq 50)$

CASING MODAL DAMPING

1	ZETAC(1)	
13	⋮	
25	⋮	6E12.8
37	⋮	
49	⋮	
61	ZETAC(NCMØD)	
	IDENTIFICATION 73	6069 XX 80

ZETAC(I) = MODAL DAMPING VALUE TO BE APPLIED TO EACH CASING MODE, IN TERMS OF CRITICAL DAMPING

(i.e. 2% DAMPING WOULD BE ENTERED AS ZETAC = 0.02)

NCMØD ENTRIES REQUIRED

FUNCTION GENERATOR DATA

1	NPT	2I12
13	IPLT	
25	TYTL	4A10, 1A8
37		
49		
61		
IDENTIFICATION 73	607XX	80

NPT = NUMBER OF POINTS IN THIS FUNCTION GENERATOR (≤ 25)

IPLT = 0 NO PLOT
= 1 CREATE A PLOT OF THIS FUNCTION

TYTL = TITLE TO BE PLACED AT THE TOP OF THE PLOT OF THIS FUNCTION – MAY OCCUPY COLUMNS 25–72, MUST END WITH $

1	SPEED(I)	
13	FG(I)	
25	⋮	6E12.8
37	⋮	
49	SPEED(NPT)	
61	FG(NPT)	
IDENTIFICATION 73	607XX 01	80

PREPARED BY:	Rocketdyne Division Rockwell International	PAGE NO. 13
CHECKED BY:		REPORT NO.
DATE:		

FUNCTION GENERATOR DATA (CONT'D)

SPEED(I) = ROTOR SPIN SPEED OF I TH POINT IN THIS FUNCTION

FG (I) = FUNCTION VALUE OF I TH POINT

REPEAT THIS CARD UNTIL NPT POINTS HAVE BEEN DESCRIBED - ALL FUNCTIONS SHOULD BE DEFINED OVER THE ENTIRE SPEED RANGE COVERED IN THIS RUN AS DESCRIBED BY CARD 6060

REPEAT THIS SET OF CARDS UNTIL NFGEN SETS ARE ENTERED. FUNCTION GENERATORS ARE INDEXED 1 THROUGH NFGEN IN THE ORDER THEY ARE READ.

FORM R 152-R-2 REV. 10-73

NSL ADDITIONS

DEFINES STIFFNESS AND DAMPING ADDITIONS CONNECTING GROUPS

1	I	RR	IRC	ICR	ICC	8I3
13		L1	L2	L3	L4	
25		CS(1)				4E12.8
37		CS(2)				
49		CS(3)				
61		CS(4)				
	IDENTIFICATION	73	6080	1.0	80	

THIS CARD DESCRIBES THE STIFFNESS ADDITIONS FOR THE ELEMENTAL MATRIX BETWEEN TWO POINTS

$$\begin{Bmatrix} F_Y \\ F_Z \end{Bmatrix} = \left[\begin{array}{c|c} CS(1) & CS(3) \\ \hline CS(4) & CS(2) \end{array}\right] \begin{Bmatrix} Y \\ Z \end{Bmatrix}$$

1	K1	K2	K3	K4	4I3
13	CD(1)				
25	CD(2)				4E12.8
37	CD(3)				
49	CD(4)				
61					
IDENTIFICATION	73	6080	1.1	80	

THIS CARD DESCRIBES THE DAMPING ADDITIONS CONNECTING THE SAME POINTS

$$\begin{Bmatrix} F_Y \\ F_Z \end{Bmatrix} = \left[\begin{array}{c|c} CD(1) & CD(3) \\ \hline CD(4) & CD(2) \end{array}\right] \begin{Bmatrix} \dot{Y} \\ \dot{Z} \end{Bmatrix}$$

NSL ADDITIONS (CONTINUED)

IRR = ROTOR GROUP Y DOF
IRC = ROTOR GROUP Z DOF
ICR = CASING GROUP Y DOF
ICC = CASING GROUP Z DOF

L1 = FUNCTION GENERATOR NUMBER OPERATING ON COEFFICIENT CS(1)
L2 = FUNCTION GENERATOR NUMBER OPERATING ON COEFFICIENT CS(2)
L3 = FUNCTION GENERATOR NUMBER OPERATING ON COEFFICIENT CS(3)
L4 = FUNCTION GENERATOR NUMBER OPERATING ON COEFFICIENT CS(4)

K1, K2, K3, K4 = FUNCTION GENERATOR NUMBER OPERATING ON DAMPING COEFFICIENTS CD(1), CD(2), CD(3), CD(4) RESPECTIVELY

$\boxed{\text{NSL PAIRS OF CARDS REQUIRED}}$

GENERAL ADDITIONS

1	IR	IC	L1	L2	4I3
13	S				2E12.8
25	D				
37					
49					
61					
IDENTIFICATION	73	6091			80

THESE CARDS DEFINE GENERAL STIFFNESS AND DAMPING ADDITIONS EITHER BETWEEN ROTOR AND CASING, WITHIN THE ROTOR ONLY, OR WITHIN THE CASING ONLY.

<u>S</u> = STIFFNESS ADDITION COEFFICIENT

<u>D</u> = DAMPING ADDITION COEFFICIENT

<u>IR</u> = ROW NO.

<u>IC</u> = COLUMN NO.

<u>L1</u> = FUNCTION GENERATOR NUMBER OPERATING ON COEFF. S

<u>L2</u> = FUNCTION GENERATOR NUMBER OPERATING ON COEFF. D

CARDS ARE THE SAME FORMAT FOR ALL GENERAL ADDITIONS.

GENERAL ADDITIONS (CONT'D)

ALL GENERAL ADDITION CARDS MUST BE ENTERED IN THE FOLLOWING ORDER:

PARTITION	NO. OF CARDS REQUIRED	CARD I.D.
UPPER LEFT	NSA1	6091
UPPER RIGHT	NSA2	6092
LOWER LEFT	NSA3	6093
LOWER RIGHT	NSA4	6094

UPPER LEFT ADDNS ARE FORCES ON THE ROTOR DUE TO MOTION OF THE ROTOR

UPPER RIGHT ADDNS ARE FORCES ON THE ROTOR DUE TO MOTION OF THE CASING

LOWER LEFT ADDNS ARE FORCES ON THE CASING DUE TO MOTION OF THE ROTOR

LOWER RIGHT ADDNS ARE FORCES ON THE CASING DUE TO MOTION OF THE CASING OR SPRINGS TO GROUND.

OUTPUT DEGREES OF FREEDOM

1	NIROT NCASE	2I6
13		
25		
37		
49		
61		
IDENTIFICATION 73	6100	80

1 CARD REQUIRED

NIROT = NO. OF ROTOR GROUP DEGREES OF FREEDOM TO APPEAR IN PRINTED OUTPUT MODE SHAPES AND/OR PLOTTED MODE SHAPES

NCASE = NO. OF CASING GROUP DEGREES OF FREEDOM TO APPEAR IN PRINTED OUTPUT MODE SHAPES AND/OR PLOTTED MODE SHAPES

NOTE: IF PLOTTED MODE SHAPES ARE TO BE REQUESTED, THE PLOTTED D.O.F.'S MUST APPEAR AT THE FRONT OF THESE LISTS, IN ORDER OF INCREASING AXIAL COORDINATE. NON-PLOTTED D.O.F.'S MAY THEN BE LISTED.

SPEED LIMITS

1	F	HIGH	2E12.8
13	F	LOW	
25			
37			
49			
61			
	IDENTIFICATION 73	6103	80

1 CARD REQUIRED

MODE SHAPE PRINT/PLOT CONTROL

ONLY MODE SHAPES WITH NATURAL FREQUENCIES GREATER THAN FLOW AND LESS THAN FHIGH WILL BE PRINTED AND/OR PLOTTED (IF REQUESTED).

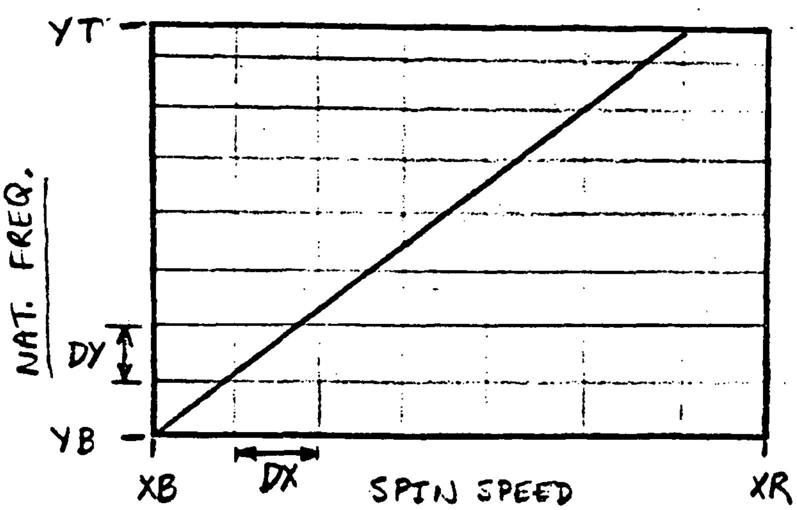
UNITS OF FHIGH AND FLOW ARE CYCLES PER MINUTE.

CRITICAL SPEED PLOT GRID DATA

Col.	Field	Format
1	XB	
13	XR	6F12.0
25	YB	
37	YT	
49	DX	
61	DY	
IDENTIFICATION 73	6104	80

1 CARD REQUIRED

THE GRID USED IN THE CRITICAL SPEED PLOT WILL BE DEFINED BY THESE PARAMETERS, AS USED BELOW.



UNITS OF XB, XR, AND DX ARE RPM.
UNITS OF YB, YT, AND DY ARE CPM.

NOTE: DY MAY BE EQUAL TO ZERO, IN WHICH CASE THE PROGRAM WILL DETERMINE AN APPROPRIATE Y-AXIS INCREMENT.

PLOTTING FLAGS

1	IPLTF I2
13	
25	
37	
49	
61	
IDENTIFICATION 73	6105 80

1 CARD REQUIRED

IPLTF = PLOTTING FLAG FOR CRITICAL SPEED MAP
=0 PLOT CRITICAL SPEEDS
=1 NO PLOT

1	IPRT2 I2
13	
25	
37	
49	
61	
IDENTIFICATION 73	6106 80

1 CARD REQUIRED

IPRT2 = PLOTTING FLAG FOR STABILITY MAP
=0 CREATE STABILITY PLOT(S)
=1 NO PLOT

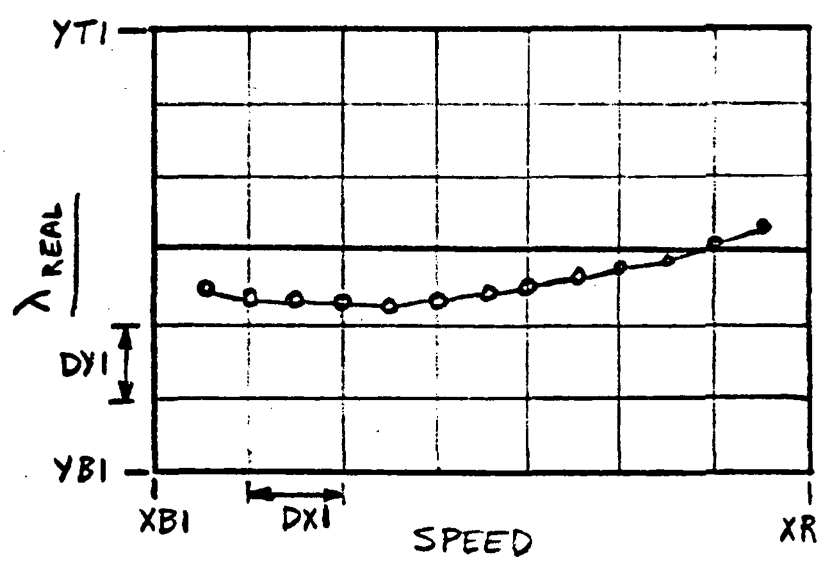
NOTE: THE FORMAT OF THE STABILITY PLOT IS DETERMINED BY KCRT

STABILITY PLOT GRID DATA

1	XB1	
13	XR1	6F12.0
25	YB1	
37	YT1	
49	DX1	
61	DY1	
	IDENTIFICATION 73	6107 80

1 CARD REQUIRED

THE GRID USED IN THE STABILITY PLOTS WILL BE DEFINED BY THESE PARAMETERS, AS DEFINED BELOW.



UNITS OF XB1, XR1, AND DX1 ARE RPM
UNITS OF YB1, YT1, AND DY1 ARE SEC^{-1}

STABILITY PLOT CHARACTERS

1	Z1	Z2	Z3	Z4	4I3
13					
25					
37					
49					
61					
IDENTIFICATION 73	6108				80

1 CARD REQUIRED

IF ONLY 4 MODES ARE TO BE PLOTTED PER FRAME, THE PROGRAMMER MAY CHOOSE, FROM THE LIST OF SYMBOLS BELOW, WHICH PLOTTING SYMBOL TO ATTACH TO EACH MODE.

Symbol	○	△	+	×	◇	▽	⊠	✳	◈	⊕	⧗	⊞	⊗	△	●	○	□	■
ZI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

NOTE: IF ALL MODES ARE PLOTTED ON ONE FRAME, SYMBOL 8 WILL BE USED FOR ALL MODES, BUT THIS CARD MUST STILL BE ENTERED.

OUTPUT DEGREES OF FREEDOM

REQUIRED ONLY IF NIROT ≠ 0

1	IROT(I)	
13		24I3
25		
37		
49		
61		
IDENTIFICATION	73 6101	80

IROT(I) = LIST OF ROTOR GROUP DOF'S TO APPEAR IN PRINTED AND/OR PLOTTED OUTPUT - NIROT ENTRIES REQUIRED

REQUIRED ONLY IF NCASE ≠ 0

1	ICASE(I)	
13		24I3
25		
37		
49		
61		
IDENTIFICATION	73 6102	80

ICASE(I) = LIST OF CASING GROUP DOF'S TO APPEAR IN PRINTED AND/OR PLOTTED OUTPUT - NCASE ENTRIES REQUIRED

MODE SHAPE PLOTTING FLAG

1	IPRT3	I2
13		
25		
37		
49		
61		
IDENTIFICATION 73	6109	80

1 CARD REQUIRED

IPRT3 = 0 PLOT SYSTEM MODE SHAPES
= 1 NO MODE SHAPE PLOTS

NOTE: IF IPRT3 = 1, THIS IS THE FINAL INPUT CARD REQUIRED

1	KRPM	I3
13		
25		
37		
49		
61		
IDENTIFICATION 73	6110	80

REQ'D ONLY IF IPRT3 = 0

KRPM = NO. OF ROTOR SPIN SPEEDS FOR WHICH MODE SHAPES WILL BE PLOTTED.

$1 \leq KRPM \leq NI$ (NO. OF SPEED INCREMENTS)

MODE SHAPE PLOTTING SPEEDS

Col	Field	Format
1	ZRPM(1)	
13	⋮	6F12.0
25	⋮	
37	⋮	
49	⋮	
61	ZRPM(KRPM)	
IDENTIFICATION 73	6111	80

ZRPM(I) = ARRAY OF SPEEDS FOR WHICH MODE SHAPES ARE TO BE PLOTTED

REPEAT THIS CARD (IF NECESSARY) UNTIL KRPM SPEEDS HAVE BEEN ENTERED

NOTE: ZRPM CAN ONLY BE EXACT SPIN SPEEDS FOR WHICH THE SYSTEM IS BEING SOLVED

ie. $ZRPM = RPM1 + DRPM * I$
$I \leq NI$

MODE SHAPE PLOTTING PARAMETERS

Col				Format
1	NSTAT	IFLG	IFLG1	I6, 2I3
13	THETA			2F12.6
25	SCALE			
37				
49				
61				
IDENTIFICATION 73	6112			80

NSTAT = NUMBER OF ROTOR/CASING STATIONS TO BE PLOTTED (≤ 10)

IFLG = FLAG TO CONTROL Y-Z DOF ORBIT PLOTS

= 1 PLOT ROTOR SHAPES ONLY
= 2 PLOT CASING SHAPES ONLY
= 3 PLOT ROTOR, CASING, AND ROTOR-CASING RELATIVE SHAPES, FOR EACH MODE

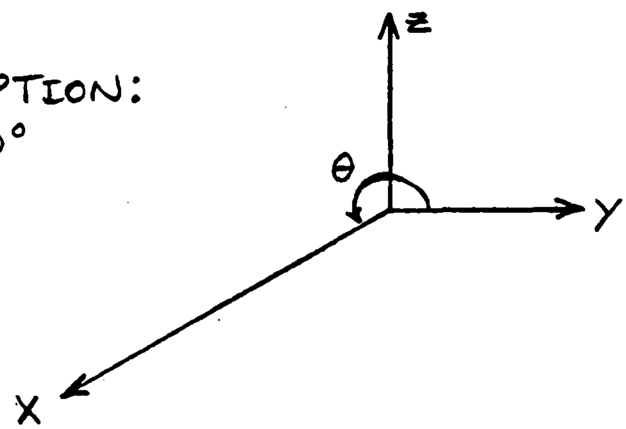
IFLG1 = FLAG FOR PLOTTING AXIAL DOF SHAPES IF X, Y, AND Z DOF'S ARE INCLUDED IN THE ROTOR MODEL

= 0 PLOT X VS. Y DEFLECTION
= 1 PLOT X VS. Z DEFLECTION
= 3 NO AXIAL DOF MODE PLOTS

PLOTTING PARAMETERS (CONT'D)

THETA = ANGLE OF ALIGNMENT OF THE X-AXIS WHEN VIEWING THE Y-Z ORBIT PLOTS (IN DEGREES)

NORMAL OPTION:
$\theta = 210°$



SCALE = A PARAMETER USED TO SCALE THE DEFLECTIONS TO AN APPROPRIATE SIZE FOR PLOTTING PURPOSES.

NORMAL OPTION: SCALE = 0.10

A SMALLER SCALE WILL INCREASE THE SIZE OF THE ORBITS, A LARGER SCALE WILL DECREASE THE SIZE OF THE ORBITS.

AXIAL COORDINATES

1	X(1)		
13	:		6F12.5
25	:		
37	:		
49	:		
61	X(NSTAT)		
IDENTIFICATION 73	6113		80

X(I) = THE X-COORDINATE OF THE ITH STATION TO BE PLOTTED. THE STATIONS MUST BE IN ORDER OF ASCENDING X-COORDINATE.
(MAX. OF 10)

ie. X(I+1) > X(I)

REPEAT THIS CARD (IF NECESSARY) UNTIL NSTAT ENTRIES ARE MADE.

NOTE: REMEMBER THAT THE DOF'S IN IROT(I) AND ICASE(I) MUST CORRESPOND TO THE COORDINATES IN X(I).

```
*PRERSTAB***************************************************************
C
C     P R O G R A M   P R E R S T A B
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C***********************************************************************
C
C     PRINCIPAL SUBROUTINES UTILIZED IN PRERSTAB ARE CALLED
C     IN THE FOLLOWING ORDER:
C
C     FORT01 . . . . . . . . . . . . . . READ PROBLEM SIZE DATA, READ AND
C                                        NORMALIZE GYROSCOPIC ADDITIONS
C             R1
C                      MODAL
C             TRANS
C
C     FORT02 . . . . . . . . . . . . . . READ SUBCASE DATA, MATRIX DATA
C
C             ADDS
C             ADDM
C             OUTAB
C             R3
C
C***********************************************************************
C
C     INPUT FILE NAMES    DESCRIPTION
C     ----------------    ----------------------------------------------
C
C     <USER-DEFINEABLE>   FORMATTED USER'S INPUT FILE, INCLUDING ANY
C                         SUBCASE INPUT AND SUBSTRUCTURE MODE SHAPES
C
C     <USER-DEFINEABLE>   FORMATTED ROTOR NORMAL MODES (IF NOT INCLUDED
C                         IN THE USER'S INPUT FILE)
C
C     <USER-DEFINEABLE>   FORMATTED CASING NORMAL MODES (IF NOT INCLUDED
C                         IN THE USER'S INPUT FILE)
C
C
C     OUTPUT FILE NAMES   DESCRIPTION
C     -----------------   ----------------------------------------------
C
C     <USER-DEFINEABLE>   FORMATTED OUTPUT LISTING.  CAN BE 'PRN' (FOR
C                         IMMEDIATE PRINTING) OR ANY DISK FILE
C
C     RUNDATA.BIN         BINARY RUN DATA FILE FOR USE BY PROGRAM RSTAB
C
C     FGPLTS.BIN          BINARY FUNCTION GENERATOR DATA FOR PLOTTING BY
C                         PROGRAM PSTRSTAB
C
C     FIN.BAT             FORMATTED IBM BATCH FILE TO COPY OUTPUT
C                         LISTING FILES CREATED BY PROGRAMS RSTAB AND
C                         PSTRSTAB (IF NOT PRINTED IMMEDIATELY) TO THE
C                         USER-DEFINED OUTPUT LISTING FILE
C
C     LOG.SAV             BINARY USAGE LOG FOR PRODUCTIVITY ACCOUNTING
C
C***********************************************************************
$STORAGE:2
```

```
C
      PROGRAM FRERSTAB
C
      CHARACTER*1  TITL(72,2),C1,C2
      CHARACTER    BANNR(3)*31,LOGFIL*14,FIN*14,FOUT*14
C
      INTEGER      NSA(4),NZ(4),NDATE(3),NTIME(4)
C
      REAL         RZ(6),RZ1(6)
C
      LOGICAL      PRN
C
      EQUIVALENCE (C1,FOUT),(C2,LOGFIL)
C
C* ARRAY DIMENSIONS MOST LIKELY TO CHANGE ****************************
C
C-----NOTES: DIMENSION WK TO LRDOF
C            DIMENSION G TO (LRDOF,LRDOF)
C
      REAL*8       GBAR(16,16),WK(115)
C
      INTEGER      JFUN(160,2),NPT(24),NRC(100,2,4),IROT(20),ICASE(20)
C
      DIMENSION    AC(50,14),AR(115,16),WC(14),WR(16),ZETAC(14),
     +             SPEED(20,24),DMP(100,4),S(100,4),FG(20,24),
     +             G(115,115),ZRPM(25),X(10)
C
C*********************************************************************
C
      COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
     +                 KRPM,NFGEN,NSA,NSL,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
     +                 NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,IFLG1,
     +                 THETA,SCALE,MAXPTS,IPCNT,ISCNT
C
      COMMON    /MEM/G
C
      DATA BANNR /'P R O G R A M   R S T A B',
     +            'IBM PC VERSION 1.0',
     +            'APRIL 1984'/
      DATA LOGFIL/' :LOG.SAV'/
C
C*****************************************************************
C
C-----ARRAY DIMENSIONS
C
      DATA LCMOD,LRMOD/14,16/,          LSA,LSJ    /100,220/
      DATA LPT        /20/,             LRDOF,LCDOF/115,50/
C
C******************************************************************
C
      LG=LRDOF
C
C-----SET UP SCREEN
C
      CALL QCLEAR(1,7)
      CALL QBORD(1)
      CALL QCSIZ(0,0)
      CALL QCMOV(0,24)
C
```

```
C-----PROGRAM BANNER
C
      CALL QDATE(NDATE(3),NDATE,NDATE(2))
      CALL QTIME(NTIME,NTIME(2),NTIME(3),NTIME(4))
      NTIME(3) = NTIME(3) + NTIME(4)/100. + 0.5
C
      WRITE(*,8004) BANNR
      WRITE(*,8005) (NDATE(I),I=1,3),(NTIME(I),I=1,3)
C
C-----OPEN OUTPUT LISTING FILE
C
      CALL QCSIZ(0,7)
      WRITE(*,8001) 'ENTER OUTPUT LISTING FILE NAME: '
      READ(*,9000) FOUT
      IF(FOUT.EQ.' ') FOUT='PRN'
    5 IF(C1.EQ.' ') THEN
        READ(FOUT,'(1X,A)') FOUT
        GOTO 5
      ENDIF
      OPEN(6,FILE=FOUT,STATUS='NEW')
      WRITE(6,8004) BANNR
      WRITE(6,8005) (NDATE(I),I=1,3),(NTIME(I),I=1,3)
C
C-----CREATE WRAP-UP BATCH FILE
C
      OPEN(2,FILE='FIN.BAT',STATUS='NEW')
      PRN=FOUT.EQ.'PRN'.OR.FOUT.EQ.'LPT1:'.OR.
     +    FOUT.EQ.'prn'.OR.FOUT.EQ.'lpt1:'
      IF(.NOT.PRN) WRITE(2,8002) FOUT
      CLOSE(2)
C
C-----OPEN INPUT FILE
C
      WRITE(*,8001) 'ENTER FILE NAME FOR INPUT DATA: '
      READ(*,9000) FIN
      OPEN(5,FILE=FIN,STATUS='OLD')
C
C-----OPEN RUN DATA FILE AND FUNCTION GEN PLOTTING FILE
C
      OPEN(1,FILE='RUNDATA.BIN',STATUS='NEW',FORM='UNFORMATTED')
      OPEN(2,FILE='FGPLTS.BIN',STATUS='NEW',FORM='UNFORMATTED')
C
C-----OPEN LOG FILE
C
      WRITE(*,8001) 'ENTER DRIVE CONTAINING THE PROGRAM USAGE LOG: '
      READ(*,9000) C2
      OPEN(7,FILE=LOGFIL,STATUS='OLD',ACCESS='DIRECT',
     +       FORM='UNFORMATTED',RECL=242)
      READ(7,REC=1) NRECS
C
C-----F O R T O 1
C
      NXTC=4
      IPCNT=0
C
   15 IPCNT=IPCNT+1
      ISCNT=0
C
      CALL QCLEAR(1,7)
```

```
      CALL QCMOV(0,24)
C
      CALL FORT01(AC,AR,G,LG,GBAR,LCDOF,LRDOF,LRMOD,WK,WC,WR)
C
C-----SAVE RESTART DATA
C
      WRITE(1) (TITL(I,1),I=1,72),NCDOF,NCMOD,NRDOF,NRMOD,IPCNT
C
      WRITE(1) ((AC(I,J),I=1,NCDOF),J=1,NCMOD),((AR(I,J),I=1,NRDOF),
     +         J=1,NRMOD),((GBAR(I,J),I=1,NRMOD),J=1,NRMOD),(WC(I),
     +         I=1,NCMOD),(WR(I),I=1,NRMOD)
C
C----- F O R T 0 2
C
   20 ISCNT=ISCNT+1
C
      CALL QCLEAR(1,7)
      CALL QCMOV(0,24)
C
      WRITE(2) IPCNT,ISCNT
C
      CALL FORT02(ZETAC,FG,SPEED,NPT,NRC,S,DMP,JFUN,LPT,LSA,LSJ,
     +            ZRPM,ICASE,IROT,X,NZ,RZ,RZ1)
C
C-----SAVE SUBCASE DATA
C
      WRITE(1) (TITL(I,2),I=1,72),NSA,NSL,IPRT,KCRT,NXTC,NFGEN,MAXPTS,
     +         RPM1,DRPM,NI,NROT,NCASE,FHIGH,FLOW,RZ,IPLTF,IPRT2,RZ1,
     +         NZ,IPRT3,KRPM,NSTAT,IFLG,IFLG1,THETA,SCALE,ISCNT
C
      NSJ=4*NSL+NSA(1)+NSA(2)+NSA(3)+NSA(4)
      NNSA=4*NSL+MAX0(NSA(1),NSA(2),NSA(3),NSA(4))
C
      WRITE(1) (ZETAC(I),I=1,NCMOD),(NPT(J),(FG(I,J),SPEED(I,J),
     +         I=1,MAXPTS),J=1,NFGEN),((JFUN(I,J),I=1,NSJ),J=1,2),
     +         ((S(I,J),DMP(I,J),(NRC(I,K,J),K=1,2),I=1,NNSA),J=1,4),
     +         (IROT(I),I=1,NROT),(ICASE(I),I=1,NCASE),(ZRPM(I),
     +         I=1,KRPM),(X(I),I=1,NSTAT)
C
C-----UPDATE LOG FILE
C
      WRITE(7,REC=NRECS+1) NDATE,NTIME,TITL,FIN,FOUT,NGYRO,NCMOD,NRMOD,
     +                     NCDOF,NRDOF,NCASE,NROT,KRPM,NFGEN,MAXPTS,NSA,
     +                     NSL,NSJ,NNSA,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
     +                     NXTC,NI,NSTAT,IFLG,IPCNT,ISCNT
      NRECS=NRECS+1
C
C-----RERUN?
C
      GOTO(15,20,20) NXTC
      CALL QCSIZ(0,0)
C
C-----CLOSE FILES
C
      CLOSE(1)
      CLOSE(2)
      CLOSE(7)
C
C-----FORMAT STATEMENTS
```

```
C
 8001 FORMAT(//2X,A,1X\)
 8002 FORMAT('COPY ',A,'+LIST2+LIST3'/'DEL LIST2'/'DEL LIST3')
 8004 FORMAT(1H1,5(/),2X,78(1H*),3(/7H  *****,68X,5H*****/7H  *****,18X,
     +          A,19X,5H*****)/7H  *****,68X,5H*****/2X,78(1H*))
 8005 FORMAT(2X,5HDATE:,I3,1H/,I2,1H/,I4,42X,11HBEGIN TIME:,I3,
     +          2(1H:,I2),6(/))
 9000 FORMAT(72A)
C
      END
```

Name	Type	Offset	P	Class
AC	REAL	12886		
AR	REAL	15686		
BANNR	CHAR*31	12792		
C1	CHAR*1	1238		
C2	CHAR*1	1252		
DMP	REAL	9272		
DRPM	REAL	188		/DATA /
FG	REAL	10872		
FHIGH	REAL	194		/DATA /
FIN	CHAR*14	23070		
FLOW	REAL	198		/DATA /
FOUT	CHAR*14	1238		
G	REAL	0		/MEM /
GBAR	REAL*8	6504		
I	INTEGER*2	23062		
ICASE	INTEGER*2	9232		
IFLG	INTEGER*2	204		/DATA /
IFLG1	INTEGER*2	206		/DATA /
IPCNT	INTEGER*2	218		/DATA /
IPLTF	INTEGER*2	172		/DATA /
IPRT	INTEGER*2	178		/DATA /
IPRT2	INTEGER*2	174		/DATA /
IPRT3	INTEGER*2	176		/DATA /
IROT	INTEGER*2	9192		
ISCNT	INTEGER*2	220		/DATA /
J	INTEGER*2	23086		
JFUN	INTEGER*2	8552		
K	INTEGER*2	23104		
KCRT	INTEGER*2	180		/DATA /
KRPM	INTEGER*2	158		/DATA /
LCDOF	INTEGER*2	23058		
LCMOD	INTEGER*2	23046		
LG	INTEGER*2	23060		
LOGFIL	CHAR*14	1252		
LPT	INTEGER*2	23054		
LRDOF	INTEGER*2	23056		
LRMOD	INTEGER*2	23048		
LSA	INTEGER*2	23050		
LSJ	INTEGER*2	23052		
MAX0				INTRINSIC
MAXPTS	INTEGER*2	216		/DATA /
NCASE	INTEGER*2	154		/DATA /
NCDOF	INTEGER*2	6		/DATA /
NCMOD	INTEGER*2	2		/DATA /
NDATE	INTEGER*2	4850		
NFGEN	INTEGER*2	160		/DATA /

```
D Line# 1       7
NGYRD   INTEGER*2        0   /DATA   /
NI      INTEGER*2      192   /DATA   /
NNSA    INTEGER*2    23102
NPT     INTEGER*2     4856
NRC     INTEGER*2     4904
NRDOF   INTEGER*2        8   /DATA   /
NRECS   INTEGER*2    23084
NRMOD   INTEGER*2        4   /DATA   /
NROT    INTEGER*2      156   /DATA   /
NSA     INTEGER*2      162   /DATA   /
NSJ     INTEGER*2    23100
NSL     INTEGER*2      170   /DATA   /
NSTAT   INTEGER*2      202   /DATA   /
NTIME   INTEGER*2     4842
NXTC    INTEGER*2      182   /DATA   /
NZ      INTEGER*2     1266
PRN     LOGICAL*2    23068
RPM1    REAL           184   /DATA   /
RZ      REAL          1274
RZ1     REAL          1298
S       REAL          3242
SCALE   REAL           212   /DATA   /
SPEED   REAL          1322
THETA   REAL           208   /DATA   /
TITL    CHAR*1          10   /DATA   /
WC      REAL          1182
WK      REAL*8           2
WR      REAL           922
X       REAL          1142
ZETAC   REAL           986
ZRPM    REAL          1042


Name    Type          Size   Class

DATA                   222   COMMON
FORT01                       SUBROUTINE
FORT02                       SUBROUTINE
MEM                  52900   COMMON
PRERST                       PROGRAM
QBORD                        SUBROUTINE
QCLEAR                       SUBROUTINE
QCMOV                        SUBROUTINE
QCSIZ                        SUBROUTINE
QDATE                        SUBROUTINE
QTIME                        SUBROUTINE

  Pass One     No Errors Detected
             246 Source Lines
```

```
*FORT01.FOR*************************************************************
C
C      S U B R O U T I N E   F O R T 0 1
C
C***********************************************************************
C
C      SUBROUTINES UTILIZED IN FORT01 ARE CALLED
C      IN THE FOLLOWING ORDER:
C
C      R1. . . . . . . READ THE MODE SHAPES AND NATURAL FREQUENCIES
C
C      TRANS . . . . . TRANSFORM THE GYROSCOPIC ADDITIONS
C                       INTO NORMAL COORDINATES
C
C***********************************************************************
$STORAGE:2
C
      SUBROUTINE FORT01(AC,AR,G,LG,GBAR,LCDOF,LRDOF,LRMOD,WK,WC,WR)
C
      CHARACTER  TITL(72,2),TAPE4*14,LINE1*72,LINE2*72
C
      EQUIVALENCE (TITL,LINE1)
C
      DIMENSION  AC(LCDOF,1),AR(LRDOF,1),WC(1),WR(1),G(LG,1)
C
      REAL*8     GBAR(LRMOD,1),WK(1)
C
      COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
     +                  KRPM,NPGEN,NSA(4),NSL,IPLTF,IPRT2,IPRT3,IPRT,
     +                  KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,
     +                  IFLG1,THETA,SCALE,MAXPTS,IPCNT,ISCNT
C
C-----PROBLEM HEADER
C
      WRITE(6,8002) IPCNT
      WRITE(*,8002) IPCNT
C
C-----READ TITLE 1 AND FIRST INPUT DATA LINE.
C      SKIP THROUGH ANY PREDEEDING JCL.
C
      READ(5,9000,END=80) LINE2
C
   10 LINE1=LINE2
      READ(5,9000,END=80) LINE2
      READ(LINE2,8001,ERR=10) NRDOF,NRMOD,NCDOF,NCMOD,NGYRO,NRTAP,NCTAP,
     +                         GRAV,MLIST
C
      WRITE(6,6) (TITL(I,1),I=1,72)
      WRITE(6,8000) NRDOF,NRMOD,NCDOF,NCMOD,NRTAP,NCTAP,NGYRO,GRAV
      WRITE(*,6) (TITL(I,1),I=1,72)
      WRITE(*,8000) NRDOF,NRMOD,NCDOF,NCMOD,NRTAP,NCTAP,NGYRO,GRAV
C
C-----GET FILE NAMES FOR MODAL DATA
C
      IF(NRTAP.NE.0) THEN
        WRITE(*,8003) 'ENTER THE ROTOR MODE SHAPE FILE NAME:'
      READ(*,9000) TAPE4
      OPEN(4,FILE=TAPE4,STATUS='OLD')
      ENDIF
```

```
C
      IF(NCTAP.NE.0) THEN
        WRITE(*,8003) 'ENTER THE CASING MODE SHAPE FILE NAME:'
      READ(*,9000) TAPE4
      OPEN(3,FILE=TAPE4,STATUS='OLD')
      ENDIF
C
C-----READ AND PRINT ROTOR AND CASING MODAL
C     MATRICES, NATURAL FREQUENCIES, AND
C     GYROSCOPIC ELEMENTS
C
      CALL R1(AC,AR,G,LG,LCDOF,LRDOF,NCTAP,NRTAP,WC,WR,GRAV,MLIST,TAPE4)
C
C-----CLOSE THE MODAL DATA FILES
C
      IF(NRTAP.NE.0) CLOSE(4)
      IF(NCTAP.NE.0) CLOSE(3)
C
C-----ZERO THE GBAR MATRIX
C
      DO 30 I=1,NRMOD
      DO 30 J=1,NRMOD
   30 GBAR(I,J)=0.0D0
      IF(NGYRO.EQ.0) RETURN
C
C-----FORM MODAL GYROSCOPIC MATRIX GBAR
C
      CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AR,LRDOF,NRDOF,NRMOD,
     +           GBAR,LRMOD,WK)
C
      DO 70 I = 1,NRMOD
      DO 70 J = 1,NRMOD
   70 GBAR(I,J) = GBAR(I,J) / GRAV
C
      RETURN
C
   80 WRITE(*,*) 'ERROR READING INPUT FILE'
      STOP
C
C-----FORMAT STATEMENTS
C
    6 FORMAT(2X,72A///)
 8000 FORMAT(5X,'NO. OF ROTOR DOF  =',I4,5X,'NO. OF ROTOR MODES  =',I4//
     +       5X,'NO. OF CASING DOF =',I4,5X,'NO. OF CASING MODES =',I4//
     +       5X,'ROTOR TAPE CODE   =',I4,5X,'CASING TAPE CODE    =',I4//
     +       5X,'NO. OF GYROSCOPIC ADDITIONS =',I4//
     +       5X,'ACCELERATION OF GRAVITY     =',F8.2)
 8001 FORMAT(BN,7I6,E12.6,I6)
 8002 FORMAT(////5X,4('*  '),'M O D A L   I N P U T   S E C T I O N  ',
     +       I2,4('  *')////)
 8003 FORMAT(///2X,A,2X\)
 9000 FORMAT(72A)
C
      END
```

Name	Type	Offset	P	Class
AC	REAL	0	*	
AR	REAL	4	*	

```
DRPM    REAL        188    /DATA   /
FHIGH   REAL        194    /DATA   /
FLOW    REAL        198    /DATA   /
G       REAL          8 *
GBAR    REAL*8       16 *
GRAV    REAL         78
I       INTEGER*2    84
IFLG    INTEGER*2   204    /DATA   /
IFLG1   INTEGER*2   206    /DATA   /
IPCNT   INTEGER*2   218    /DATA   /
IPLTF   INTEGER*2   172    /DATA   /
IPRT    INTEGER*2   178    /DATA   /
IPRT2   INTEGER*2   174    /DATA   /
IPRT3   INTEGER*2   176    /DATA   /
ISCNT   INTEGER*2   220    /DATA   /
J       INTEGER*2   108
KCRT    INTEGER*2   180    /DATA   /
KRPM    INTEGER*2   158    /DATA   /
LCDOF   INTEGER*2    20 *
LG      INTEGER*2    12 *
LINE1   CHAR*72      10    /DATA   /
LINE2   CHAR*72       2
LRDOF   INTEGER*2    24 *
LRMOD   INTEGER*2    28 *
MAXPTS  INTEGER*2   216    /DATA   /
MLIST   INTEGER*2    82
NCASE   INTEGER*2   154    /DATA   /
NCDOF   INTEGER*2     6    /DATA   /
NCMOD   INTEGER*2     2    /DATA   /
NCTAP   INTEGER*2    76
NFGEN   INTEGER*2   160    /DATA   /
NGYRO   INTEGER*2     0    /DATA   /
NI      INTEGER*2   192    /DATA   /
NRDOF   INTEGER*2     8    /DATA   /
NRMOD   INTEGER*2     4    /DATA   /
NROT    INTEGER*2   156    /DATA   /
NRTAP   INTEGER*2    74
NSA     INTEGER*2   162    /DATA   /
NSL     INTEGER*2   170    /DATA   /
NSTAT   INTEGER*2   202    /DATA   /
NXTC    INTEGER*2   182    /DATA   /
RPM1    REAL        184    /DATA   /
SCALE   REAL        212    /DATA   /
TAPE4   CHAR*14      88
THETA   REAL        208    /DATA   /
TITL    CHAR*1       10    /DATA   /
WC      REAL         36 *
WK      REAL*8       32 *
WR      REAL         40 *
```

```
*R1.FOR*******************************************************************
C
C           R1  READS AND WRITES THE MODAL INPUT DATA
C
C*************************************************************************
C
      SUBROUTINE R1(AC,AR,G,LG,LCDOF,LRDOF,NCTAP,NRTAP,WC,WR,GRAV,MLIST,
     +                 LINE1)
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.13 8/05/83
          C
                 DIMENSION  AC(LCDOF,1),AR(LRDOF,1),WC(1),WR(1),G(LG,1)
          C
                 CHARACTER  LINE1*14,TITL(72,2)*1
          C
                 COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
               +                  KRPM,NFGEN,NSA(4),NSL,IPLTF,IPRT2,IPRT3,IPRT,
               +                  KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,
               +                  IFLG1,THETA,SCALE,MAXPTS,IPCNT,ISCNT
          C
          C-----READ AND WRITE THE MODAL MATRICES
          C
                 IF(MLIST.EQ.0) WRITE(6,5)
                 CALL MODAL(AR,LRDOF,NRDOF,NRMOD,4,WR,GRAV,NRTAP,MLIST,LINE1)
          C
                 IF(MLIST.EQ.0) WRITE(6,10)
                 CALL MODAL(AC,LCDOF,NCDOF,NCMOD,3,WC,GRAV,NCTAP,MLIST,LINE1)
          C
          C-----READ AND WRITE THE NATURAL FREQUENCIES
          C
                 GOTO(710,711) NRTAP
          C
                 READ(5,6) (WR(I),I=1,NRMOD)
                 GO TO 711
          C
             710 READ(4,'(A)') LINE1
                 IF(LINE1.NE.'FREQUENCIES') GOTO 710
                 READ(4,6) (WR(I),I=1,NRMOD)
          C
             711 CONTINUE
                 WRITE(6,24)
                 WRITE(6,305)
          C
                 DO 320 I = 1,NRMOD
  1          320 WRITE(6,80) WR(I),WR(I)/6.2831853
          C
                 GOTO(720,721) NCTAP
          C
                 READ(5,6) (WC(I),I=1,NCMOD)
                 GOTO 721
          C
             720 READ(3,'(A)') LINE1
                 IF(LINE1.NE.'FREQUENCIES') GOTO 720
                 READ(3,6) (WC(I),I=1,NCMOD)
          C
             721 CONTINUE
                 WRITE(6,25)
                 WRITE(6,305)
          C
                 DO 330 I = 1,NCMOD
  1          330 WRITE(6,80) WC(I),WC(I)/6.2831853
                 IF(NGYRO.EQ.0) RETURN
          C
          C-----ZERO THE GYROSCOPIC MATRIX
          C
                 DO 340 I=1,NRDOF
  1              DO 340 J=1,NRDOF
  2          340 G(I,J)=0.0
          C
```

```
      C-----READ GYROSCOPIC MATRIX ELEMENTS
      C
              WRITE(6,27)
      C
              DO 40 IN=1,NGYRO,2
1                READ(5,30)  I,J,G(I,J),K,L,G(K,L)
1          40    WRITE(6,26) I,J,G(I,J),K,L,G(K,L)
      C
      C-----FORMAT STATEMENTS
      C
            5 FORMAT(1H1//5X,'* * *  ROTOR GROUP MODAL MATRIX   * * *'/)
            6 FORMAT(BN,6E12.6)
           10 FORMAT(1H1//5X,'* * *  CASING GROUP MODAL MATRIX   * * *'/)
           24 FORMAT(1H1//5X,'* * *  ROTOR GROUP NATURAL FREQUENCIES  -  WR   * *
             1 *',//)
           25 FORMAT(////5X,'* * *  CASING GROUP NATURAL FREQUENCIES  -  WC   * *
             1 *',//)
           26 FORMAT(2(2I8,1P1E20.8,1X))
           27 FORMAT(1H1//5X,'* * *  GYROSCOPIC MATRIX ELEMENTS   * * *'///
             1         2(5X,'ROW',5X,'COL',10X,'VALUE',6X)/)
           30 FORMAT(BN,2I6,E12.8)
           80 FORMAT(10X,1PE15.5,7X,1PE15.5)
          305 FORMAT(16X,'RAD/SEC',16X,'HERTZ',/)
      C
              END
```

Name	Type	Offset	P	Class
AC	REAL	0	*	
AR	REAL	4	*	
DRPM	REAL	188		/DATA /
FHIGH	REAL	194		/DATA /
FLOW	REAL	198		/DATA /
G	REAL	8	*	
GRAV	REAL	40	*	
I	INTEGER*2	622		
IFLG	INTEGER*2	204		/DATA /
IFLG1	INTEGER*2	206		/DATA /
IN	INTEGER*2	652		
IPCNT	INTEGER*2	218		/DATA /
IPLTF	INTEGER*2	172		/DATA /
IPRT	INTEGER*2	178		/DATA /
IPRT2	INTEGER*2	174		/DATA /
IPRT3	INTEGER*2	176		/DATA /
ISCNT	INTEGER*2	220		/DATA /
J	INTEGER*2	644		
K	INTEGER*2	660		
KCRT	INTEGER*2	180		/DATA /
KRPM	INTEGER*2	158		/DATA /
L	INTEGER*2	662		
LCDOF	INTEGER*2	16	*	
LG	INTEGER*2	12	*	
LINE1	CHAR*14	48	*	
LRDOF	INTEGER*2	20	*	
MAXPTS	INTEGER*2	216		/DATA /
MLIST	INTEGER*2	44	*	
NCASE	INTEGER*2	154		/DATA /
NCDOF	INTEGER*2	6		/DATA /
NCMOD	INTEGER*2	2		/DATA /

```
D Line# 1      7
NCTAP   INTEGER*2      24 *
NFGEN   INTEGER*2     160   /DATA   /
NGYRO   INTEGER*2       0   /DATA   /
NI      INTEGER*2     192   /DATA   /
NRDOF   INTEGER*2       8   /DATA   /
NRMOD   INTEGER*2       4   /DATA   /
NROT    INTEGER*2     156   /DATA   /
NRTAP   INTEGER*2      28 *
NSA     INTEGER*2     162   /DATA   /
NSL     INTEGER*2     170   /DATA   /
NSTAT   INTEGER*2     202   /DATA   /
NXTC    INTEGER*2     182   /DATA   /
RPM1    REAL          184   /DATA   /
SCALE   REAL          212   /DATA   /
THETA   REAL          208   /DATA   /
TITL    CHAR*1         10   /DATA   /
WC      REAL           32 *
WR      REAL           36 *
```

```
      *MODAL.FOR*****************************************************************
      C
      C          MODAL  READS MODAL INFORMATION, I.E.MODE SHAPES,
      C                 FROM STARDYNE TAPE4.
      C
      C*************************************************************************
      C
            SUBROUTINE MODAL(AR,LRDOF,NRDOF,NRMOD,NDEV,WR,GRAV,NRTAP,MLIST,
           +                      LINE1)
      C
            CHARACTER  T*4,LINE1*12
      C
            DIMENSION  AR(LRDOF,1),B(2),WR(1)
      C
            GOTO(400,600,407) NRTAP
      C
            DO 300 I=1,NRMOD
1       300 READ(5,6)  (AR(J,I),J=1,NRDOF)
            GO TO 420
      C
        400 READ(NDEV,'(A)') LINE1
            IF(LINE1.NE.' V9568 MODES') GOTO 400
      C
            DO 401 I=1,NRMOD
1           READ(NDEV,'(/)')
1       401 READ(NDEV,6)  (AR(J,I),J=1,NRDOF)
            GO TO 420
      C
        407 READ(NDEV,6)  ((AR(J,I),J=1,NRDOF),I=1,NRMOD)
            GO TO 420
      C
        600 READ(NDEV,'(A)') T
            IF(T .NE. 'DISP') GO TO 600
      C
            READ(NDEV,110) MODE,FREQ,GENWT
            CF=1.0/SQRT(GENWT/GRAV)
            WR(MODE) = FREQ * 6.2831853
      C
         15 READ(NDEV,120) J,I,B
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.13 8/05/83
               IF(J .LT. 1) GO TO 20
               IDOF=4*(J-1)+I
               IF(I .EQ. 4) IDOF=IDOF-1
               AR(IDOF,MODE)=B(1)*CF
               AR(IDOF+1,MODE)=B(2)*CF
               GO TO 15
          C
            20 IF(MODE .NE. NRMOD) GO TO 600
          C
           420 IF(MLIST.NE.0) RETURN
          C
          C-----LINE COUNTER
          C
               LCOUNT=4
          C
               DO 605 I=1,NRMOD
1                LCOUNT=LCOUNT+NRDOF/5.+4.9
1                IF(LCOUNT.GT.63) THEN
1                  WRITE(6,'(1H1)')
1                  LCOUNT=NRDOF/5.+5.9
1                ENDIF
1          605 WRITE(6,8) I,(AR(J,I),J=1,NRDOF)
          C
             6 FORMAT(BN,6E12.6)
             8 FORMAT(//5X,'MODE NO.',I6//(3X,1P5E15.5))
           110 FORMAT(I3,F17.0,F13.0)
           120 FORMAT(2X,I9,I1,20X,2F20.0)
          C
               END
```

Name	Type	Offset	P	Class
AR	REAL	0	*	
B	REAL	1140		
CF	REAL	1182		
FREQ	REAL	1174		
GENWT	REAL	1178		
GRAV	REAL	24	*	
I	INTEGER*2	1148		
IDOF	INTEGER*2	1186		
J	INTEGER*2	1156		
LCOUNT	INTEGER*2	1188		
LINE1	CHAR*12	36	*	
LRDOF	INTEGER*2	4	*	
MLIST	INTEGER*2	32	*	
MODE	INTEGER*2	1172		
NDEV	INTEGER*2	16	*	
NRDOF	INTEGER*2	8	*	
NRMOD	INTEGER*2	12	*	
NRTAP	INTEGER*2	28	*	
SQRT				INTRINSIC
T	CHAR*4	1168		
WR	REAL	20	*	

```
     *TRANS.FOR*****************************************************************
   C
   C      S U B R O U T I N E   T R A N S
   C
```

```
D Line# 1      7                               Microsoft FORTRAN77 V3.13 8/05/83
      C**********************************************************************
      C
      C      PURPOSE:      MATRIX TRANSFORMATION DEFINED AS:
      C
      C                                          T
      C                      [D] =  [D]  -  [A]  *  [G]  *  [C]
      C
      C
      C      LATEST REV:   MARCH 21, 1984
      C
      C      USAGE:        CALL TRANS(A,LRA,NRA,NCA,G,LRG,C,LRC,NRC,NCC,
      C                     +          D,LRD,WK)
      C
      C      ARGUMENTS:    A       - MATRIX TO BE TRANSPOSED AND USET TO
      C                                PRE-MULTIPLY MATRIX B.
      C                      LRA     - ROW DIMENSION OF MATRIX A EXACTLY AS
      C                                DEFINED IN CALLING PROGRAM.
      C                      NRA     - ROW ORDER OF MATRIX A (NUMBER OF ROWS
      C                                USED).
      C                      NCA     - COLUMN ORDER OF MATRIX A (NUMBER OF
      C                                COLUMNS USED).
      C                      G       - THE MATRIX OPERATED ON.
      C                      LRG     - ROW DIMENSION OF MATRIX G EXACTLY AS
      C                                DEFINED IN CALLING PROGRAM.
      C                      C       - MATRIX C TO BE USED TO POST-MULTIPLY
      C                                MATRIX B.
      C                      LRC     - ROW DIMENSION OF MATRIX C EXACTLY AS
      C                                DEFINED IN CALLING PROGRAM.
      C                      NRC     - ROW ORDER OF MATRIX C.
      C                      NCC     - COLUMN ORDER OF MATRIX C.
      C                      D       - MATRIX D. NOTE: D IS DOUBLE PRECISION.
      C                      LRD     - ROW DIMENSION OF MATRIX D EXACTLY AS
      C                                DEFINED IN CALLING PROGRAM.
      C                      WK      - WORK AREA DIMENSIONED TO AT LEAST NRA.
      C                                NOTE: WK IS DOUBLE PRECISION.
      C
      C**********************************************************************
      C
              SUBROUTINE TRANS(A,LRA,NRA,NCA,G,LRG,C,LRC,NRC,NCC,D,LRD,WK)
      C
              DIMENSION A(LRA,1),C(LRC,1),G(LRG,1)
      C
              REAL*8     D(LRD,1),WK(1)
      C
              DO 100 I = 1,NCC
1               DO 50 J = 1,NRA
2                 WK(J) = 0.0D0
2                 DO 50 K = 1,NRC
3          50   WK(J) = WK(J) + G(J,K)*C(K,I)
1     C
1               DO 100 L = 1,NCA
2               DO 100 M = 1,NRA
3         100 D(L,I) = D(L,I) - A(M,L)*WK(M)
      C
              END
```

```
Name    Type          Offset P Class

A       REAL               0 *
```

```
D Line# 1       7
C        REAL            24 *
D        REAL*8          40 *
G        REAL            16 *
I        INTEGER*2     1320
J        INTEGER*2     1328
K        INTEGER*2     1336
L        INTEGER*2     1344
LRA      INTEGER*2        4 *
LRC      INTEGER*2       28 *
LRD      INTEGER*2       44 *
LRG      INTEGER*2       20 *
M        INTEGER*2     1352
NCA      INTEGER*2       12 *
NCC      INTEGER*2       36 *
NRA      INTEGER*2        8 *
NRC      INTEGER*2       32 *
WK       REAL*8          48 *


Name     Type          Size   Class

DATA                    222   COMMON
FORT01                        SUBROUTINE
MODAL                         SUBROUTINE
R1                            SUBROUTINE
TRANS                         SUBROUTINE

  Pass One     No Errors Detected
              332 Source Lines
```

```
*FORT02.FOR*********************************************************************
C
C       S U B R O U T I N E   F O R T 0 2
C
C*******************************************************************************
C
C       SUBROUTINES UTILIZED IN FORT02 ARE CALLED
C       IN THE FOLLOWING ORDER:
C
C       ADDS. . . . . . READ INTERGROUP ADDITIONS
C
C       ADDM. . . . . . READ INTRAGROUP ADDITIONS
C
C       OUTAB . . . . . OUTPUT ALL ADDITIONS
C
C       R3. . . . . . . READ PLOTTING DATA
C
C*******************************************************************************
C
$STORAGE:2
C
        SUBROUTINE FORT02(ZETAC,FG,SPEED,NPT,NRC,S,DMP,JFUN,LPT,LSA,LSJ,
     +                    ZRPM,ICASE,IROT,X,NZ,RZ,RZ1)
C
        CHARACTER*1 TITL(72,2),TYTL(48)
C
        INTEGER      JFUN(LSJ,1),NPT(1),NRC(LSA,2,1),NSA(4),NZ(4),ICASE(1),
     +               IROT(1)
C
        DIMENSION    DMP(LSA,1),FG(LPT,1),S(LSA,1),SPEED(LPT,1),ZETAC(1),
     +               ZRPM(1),X(10),RZ(6),RZ1(6)
C
        COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
     +                   KRPM,NFGEN,NSA,NSL,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
     +                   NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,IFLG1,
     +                   THETA,SCALE,MAXPTS,IPCNT,ISCNT
C
C-----READ THE SUBCASE TITLE
C
        IF(NXTC.NE.3) READ(5,5) (TITL(I,2),I=1,72)
C
C-----SUBCASE HEADER
C
        WRITE(6,19) IPCNT,ISCNT,TITL
        WRITE(*,19) IPCNT,ISCNT,TITL
C
C-----READ NUMBERS OF MATRIX ADDITIONS,PRINT/PLOT FLAGS,
C       NEXT CASE FLAG,AND NUMBER OF FUNCTION GENERATORS
C
        READ(5,15) NSA,NSL,IPRT,KCRT,NXTC,NFGEN
        WRITE(6,16) (I,NSA(I),I=1,4)
        WRITE(6,17) NSL,NFGEN,IPRT,NXTC
        WRITE(*,16) (I,NSA(I),I=1,4)
        WRITE(*,17) NSL,NFGEN,IPRT,NXTC
C
C-----READ BEGINNING RPM, RPM INCREMENT,
C       AND TOTAL NUMBER OF STEPS
C
        READ(5,18) RPM1,DRPM,NI
```

```
      WRITE(6,40) RPM1,DRPM,NI
      WRITE(*,40) RPM1,DRPM,NI
C
C-----READ CASING MODAL DAMPING DATA
C
      WRITE(6,100)
      READ(5,200) (ZETAC(I),I=1,NCMOD)
      WRITE(6,300) (ZETAC(I),I=1,NCMOD)
C
C-----READ FUNCTION GENERATORS
C
      WRITE(6,50)
      MAXPTS=0
C
C     LINE COUNTER
C
      LCOUNT=6
      DO 130 I = 1,NFGEN
        READ(5,10) NPT(I),IPLT,TYTL
        READ(5,200) (SPEED(J,I),FG(J,I),J=1,NPT(I))
        MAXPTS=MAXO(MAXPTS,NPT(I))
        LCOUNT=LCOUNT+NPT(I)+5
        IF(LCOUNT.GT.63) THEN
          WRITE(6,'(1H1)')
          LCOUNT=6+NPT(I)
        ENDIF
        WRITE(6,60) I,TYTL
        WRITE(6,75) (SPEED(J,I),FG(J,I),J=1,NPT(I))
  130   IF(IPLT.EQ.1) WRITE(2) I,TYTL,TITL,NPT(I),
     +                        (SPEED(J,I),FG(J,I),J=1,NPT(I))
C
C-----WRITE END-OF-CASE RECORD ON FUNCTION GEN FILE
C
      WRITE(2) -999,TYTL,TITL,1,SPEED(1,1),FG(1,1)
C
C-----READ AND PRINT MATRIX ADDITIONS
C
      WRITE(6,22)
      IF(NSL .NE. 0) CALL ADDS(DMP,JFUN,LSA,LSJ,NRC,S)
      IF((NSA(1)+NSA(2)+NSA(3)+NSA(4)).NE.0) CALL ADDM(DMP,JFUN,
     +                                              LSA,LSJ,NRC,S)
      CALL OUTAB(DMP,JFUN,LSA,LSJ,NRC,S)
C
C-----READ THE REMAINDER OF THE INPUT FILE
C
      CALL R3(ZRPM,ICASE,IROT,X,NZ,RZ,RZ1)
C
      RETURN
C
C-----FORMAT STATEMENTS
C
    5 FORMAT(72A)
   10 FORMAT(BN,2I12,48A)
   15 FORMAT(BN,10I6)
   16 FORMAT(5X,'NUMBER OF GENERAL MATRIX ADDITIONS:'//(10X,'PARTITION',
     +       I2,' =',I4,10X,'PARTITION',I2,' =',I4))
   17 FORMAT(/5X,'NUMBER OF INTERGROUP ADDITIONS =',I4//5X,'NUMBER ',
     +       'OF FUNCTION GENERATORS  =',I4//5X,
     +       'MODE SHAPE PRINTING CODE =',I4//5X,
```

7

```
     +         'NEXT CASE CODE =',I4)
   18 FORMAT(BN,2E12.6,I12)
   19 FORMAT(1H1///5X,4('*  '),' S U B C A S E   I N P U T ',4('  *')//
     +         17X,'MODAL INPUT',I2,5X,'SUBCASE',I2,2(//5X,72A)/)
   22 FORMAT(1H1///5X,8('*  '),4X,'MATRIX   ADDITIONS',4X,8('*  '))
   40 FORMAT(/5X,'RPM1  =',1PE13.5,8X,'DRPM  =',E13.5,8X,'NI  =',I4/)
   50 FORMAT(1H1///,5X,'* * *  FUNCTION GENERATOR DATA  * * *',///)
   60 FORMAT(//5X,'FUNCTION NO.',I4,6X,48A/)
   75 FORMAT(45X,'RPM',17X,'VALUE'/(37X,1PE15.4,6X,E15.4))
  100 FORMAT(///5X,'* * * CASING GROUP MODAL DAMPING FACTORS * *,*'///)
  200 FORMAT(BN,6E12.5)
  300 FORMAT(10X,1PE14.5)
C
      END
```

Name	Type	Offset	P	Class
DMP	REAL	24	*	
DRPM	REAL	188		/DATA /
FG	REAL	4	*	
FHIGH	REAL	194		/DATA /
FLOW	REAL	198		/DATA /
I	INTEGER*2	50		
ICASE	INTEGER*2	48	*	
IFLG	INTEGER*2	204		/DATA /
IFLG1	INTEGER*2	206		/DATA /
IPCNT	INTEGER*2	218		/DATA /
IPLT	INTEGER*2	62		
IPLTF	INTEGER*2	172		/DATA /
IPRT	INTEGER*2	178		/DATA /
IPRT2	INTEGER*2	174		/DATA /
IPRT3	INTEGER*2	176		/DATA /
IROT	INTEGER*2	52	*	
ISCNT	INTEGER*2	220		/DATA /
J	INTEGER*2	64		
JFUN	INTEGER*2	28	*	
KCRT	INTEGER*2	180		/DATA /
KRPM	INTEGER*2	158		/DATA /
LCOUNT	INTEGER*2	54		
LPT	INTEGER*2	32	*	
LSA	INTEGER*2	36	*	
LSJ	INTEGER*2	40	*	
MAXO				INTRINSIC
MAXPTS	INTEGER*2	216		/DATA /
NCASE	INTEGER*2	154		/DATA /
NCDOF	INTEGER*2	6		/DATA /
NCMOD	INTEGER*2	2		/DATA /
NFGEN	INTEGER*2	160		/DATA /
NGYRO	INTEGER*2	0		/DATA /
NI	INTEGER*2	192		/DATA /
NPT	INTEGER*2	12	*	
NRC	INTEGER*2	16	*	
NRDOF	INTEGER*2	8		/DATA /
NRMOD	INTEGER*2	4		/DATA /
NROT	INTEGER*2	156		/DATA /
NSA	INTEGER*2	162		/DATA /
NSL	INTEGER*2	170		/DATA /
NSTAT	INTEGER*2	202		/DATA /
NXTC	INTEGER*2	182		/DATA /

```
D Line# 1      7
NZ      INTEGER*2        60 *
RPM1    REAL            184    /DATA   /
RZ      REAL             64 *
RZ1     REAL             68 *
S       REAL             20 *
SCALE   REAL            212    /DATA   /
SPEED   REAL              8 *
THETA   REAL            208    /DATA   /
TITL    CHAR*1           10    /DATA   /
TYTL    CHAR*1            2
X       REAL             56 *
ZETAC   REAL              0 *
ZRPM    REAL             44 *
```

```
      *ADDS.FOR**************************************************************
      C
      C            ADDS  READS INTERGROUP STIFFNESS AND DAMPING ADDITIONS WHICH
      C                  ACT IN THE Y AND Z DIRECTIONS. SPEED DEPENDENT FUNCTION
      C                  GENERATORS ARE ALSO ASSIGNED BY ADDS.
      C
      C**********************************************************************
      C
             SUBROUTINE ADDS(DMP,JFUN,LSA,LSJ,NRC,S)
      C
             INTEGER     IN(4),NZ(4),JFUN(LSJ,1),NRC(LSA,2,1),IQ(4),NSA(4)
      C
             DIMENSION   CD(4),CS(4),DMP(LSA,1),S(LSA,1)
      C
             CHARACTER TITL(72,2)*1
      C
             COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
            +                 KRPM,NFGEN,NSA,NSL,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
            +                 NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,IFLG1,
            +                 THETA,SCALE,MAXPTS,IPCNT,ISCNT
      C
             DATA IQ/1,-1,-1,1/
      C
             DO 300     N = 1, NSL
1            IN(1) =  4 * N  -  3
1            IN(2) =  4 * N  -  2
1            IN(3) =  4 * N  -  1
1            IN(4) =  4 * N
1     C
1            READ(5,5) NZ,(JFUN(I,1),I=4*N-3,4*N),(CS(I),I=1,4)
1     C
1            READ(5,7) (JFUN(I,2),I=4*N-3,4*N),(CD(I),I=1,4)
1     C
1            DO 10 I = 1,4
2            DO 10 II=1,4
3            S(IN(II),I)    =  CS(II) * IQ(I)
3            DMP(IN(II),I) =  CD(II) * IQ(I)
3         10 CONTINUE
1     C
1            NRC(IN(1),1,1)  =  NZ(1)
1            NRC(IN(2),1,1)  =  NZ(2)
1            NRC(IN(3),1,1)  =  NZ(1)
1            NRC(IN(4),1,1)  =  NZ(2)
1            NRC(IN(1),2,1)  =  NZ(1)
```

```
D Line# 1     7
1             NRC(IN(2),2,1)   =   NZ(2)
1             NRC(IN(3),2,1)   =   NZ(2)
1             NRC(IN(4),2,1)   =   NZ(1)
1       C
1             NRC(IN(1),1,2)   =   NZ(1)
1             NRC(IN(2),1,2)   =   NZ(2)
1             NRC(IN(3),1,2)   =   NZ(1)
1             NRC(IN(4),1,2)   =   NZ(2)
1             NRC(IN(1),2,2)   =   NZ(3)
1             NRC(IN(2),2,2)   =   NZ(4)
1             NRC(IN(3),2,2)   =   NZ(4)
1             NRC(IN(4),2,2)   =   NZ(3)
1       C
1             NRC(IN(1),1,3)   =   NZ(3)
1             NRC(IN(2),1,3)   =   NZ(4)
1             NRC(IN(3),1,3)   =   NZ(3)
1             NRC(IN(4),1,3)   =   NZ(4)
1             NRC(IN(1),2,3)   =   NZ(1)
1             NRC(IN(2),2,3)   =   NZ(2)
1             NRC(IN(3),2,3)   =   NZ(2)
1             NRC(IN(4),2,3)   =   NZ(1)
1       C
1             NRC(IN(1),1,4)   =   NZ(3)
1             NRC(IN(2),1,4)   =   NZ(4)
1             NRC(IN(3),1,4)   =   NZ(3)
1             NRC(IN(4),1,4)   =   NZ(4)
1             NRC(IN(1),2,4)   =   NZ(3)
1             NRC(IN(2),2,4)   =   NZ(4)
1             NRC(IN(3),2,4)   =   NZ(4)
1        300  NRC(IN(4),2,4)   =   NZ(3)
        C
            5 FORMAT(BN,8I3,4E12.5)
            7 FORMAT(BN,4I3,4E12.5)
        C
              END

Name    Type          Offset P Class

CD      REAL            966
CS      REAL            982
DMP     REAL              0 *
DRPM    REAL            188   /DATA   /
FHIGH   REAL            194   /DATA   /
FLOW    REAL            198   /DATA   /
I       INTEGER*2      1006
IFLG    INTEGER*2       204   /DATA   /
IFLG1   INTEGER*2       206   /DATA   /
II      INTEGER*2      1012
IN      INTEGER*2       950
IPCNT   INTEGER*2       218   /DATA   /
IPLTF   INTEGER*2       172   /DATA   /
IPRT    INTEGER*2       178   /DATA   /
IPRT2   INTEGER*2       174   /DATA   /
IPRT3   INTEGER*2       176   /DATA   /
IQ      INTEGER*2       958
ISCNT   INTEGER*2       220   /DATA   /
JFUN    INTEGER*2         4 *
KCRT    INTEGER*2       180   /DATA   /
KRPM    INTEGER*2       158   /DATA   /
```

```
D Line# 1      7
LSA     INTEGER*2         8 *
LSJ     INTEGER*2        12 *
MAXPTS  INTEGER*2       216    /DATA  /
N       INTEGER*2       998
NCASE   INTEGER*2       154    /DATA  /
NCDOF   INTEGER*2         6    /DATA  /
NCMOD   INTEGER*2         2    /DATA  /
NFGEN   INTEGER*2       160    /DATA  /
NGYRO   INTEGER*2         0    /DATA  /
NI      INTEGER*2       192    /DATA  /
NRC     INTEGER*2        16 *
NRDOF   INTEGER*2         8    /DATA  /
NRMOD   INTEGER*2         4    /DATA  /
NROT    INTEGER*2       156    /DATA  /
NSA     INTEGER*2       162    /DATA  /
NSL     INTEGER*2       170    /DATA  /
NSTAT   INTEGER*2       202    /DATA  /
NXTC    INTEGER*2       182    /DATA  /
NZ      INTEGER*2       942
RPM1    REAL            184    /DATA  /
S       REAL             20 *
SCALE   REAL            212    /DATA  /
THETA   REAL            208    /DATA  /
TITL    CHAR*1           10    /DATA  /
```

```
     *ADDM.FOR******************************************************************
     C
     C           ADDM READS INTRAGROUP STIFFNESS AND DAMPING ADDITIONS
     C           WHICH ACT IN Y AND Z DIRECTIONS. SPEED DEPENDENT FUNCTION
     C           GENERATORS ARE ALSO ASSIGNED BY ADDM.
     C
     C**************************************************************************
     C
            SUBROUTINE ADDM(DMP,JFUN,LSA,LSJ,NRC,S)
     C
            INTEGER     JFUN(LSJ,1),NRC(LSA,2,1),NSA(4)
     C
            DIMENSION  DMP(LSA,1),S(LSA,1)
     C
            CHARACTER TITL(72,2)*1
     C
            COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
           +                KRPM,NFGEN,NSA,NSL,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
           +                NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,IFLG1,
           +                THETA,SCALE,MAXPTS,IPCNT,ISCNT
     C
     C-----1. ADDITIONS  FOR  UPPER  LEFT  PARTITION
     C-----2. ADDITIONS  FOR  UPPER  RIGHT  PARTITION
     C-----3. ADDITIONS  FOR  LOWER  LEFT  PARTITION
     C-----4. ADDITIONS  FOR  LOWER  RIGHT  PARTITION
     C
            KK=0
            DO 105 II=1,4
1           IF(NSA(II) .EQ. 0) GO TO 105
1           READ(5,5) ((NRC(I,L,II),L=1,2),(JFUN(I+KK,L),L=1,2),
1          1          S(I,II),DMP(I,II),I=4*NSL+1,4*NSL+NSA(II))
1           KK=KK+NSA(II)
1       105 CONTINUE
```

```
C
      5 FORMAT(BN,4I3,2E12.6)
C
        END
```

Name	Type	Offset	P	Class
DMP	REAL	0	*	
DRPM	REAL	188		/DATA /
FHIGH	REAL	194		/DATA /
FLOW	REAL	198		/DATA /
I	INTEGER*2	1094		
IFLG	INTEGER*2	204		/DATA /
IFLG1	INTEGER*2	206		/DATA /
II	INTEGER*2	1092		
IPCNT	INTEGER*2	218		/DATA /
IPLTF	INTEGER*2	172		/DATA /
IPRT	INTEGER*2	178		/DATA /
IPRT2	INTEGER*2	174		/DATA /
IPRT3	INTEGER*2	176		/DATA /
ISCNT	INTEGER*2	220		/DATA /
JFUN	INTEGER*2	4	*	
KCRT	INTEGER*2	180		/DATA /
KK	INTEGER*2	1090		
KRPM	INTEGER*2	158		/DATA /
L	INTEGER*2	1096		
LSA	INTEGER*2	8	*	
LSJ	INTEGER*2	12	*	
MAXPTS	INTEGER*2	216		/DATA /
NCASE	INTEGER*2	154		/DATA /
NCDOF	INTEGER*2	6		/DATA /
NCMOD	INTEGER*2	2		/DATA /
NFGEN	INTEGER*2	160		/DATA /
NGYRO	INTEGER*2	0		/DATA /
NI	INTEGER*2	192		/DATA /
NRC	INTEGER*2	16	*	
NRDOF	INTEGER*2	8		/DATA /
NRMOD	INTEGER*2	4		/DATA /
NROT	INTEGER*2	156		/DATA /
NSA	INTEGER*2	162		/DATA /
NSL	INTEGER*2	170		/DATA /
NSTAT	INTEGER*2	202		/DATA /
NXTC	INTEGER*2	182		/DATA /
RPM1	REAL	184		/DATA /
S	REAL	20	*	
SCALE	REAL	212		/DATA /
THETA	REAL	208		/DATA /
TITL	CHAR*1	10		/DATA /

```
*OUTAB.FOR*****************************************************************
C
C           OUTAB  IS AN IO ROUTINE FOR STIFFNESS AND DAMPING ADDITIONS
C                  AND THEIR RELATED FUNCTION GENERATORS.
C
C           NRC,S,JFUN,DMP  COME FROM SUB. ADDM
C
C*************************************************************************
C
```

```
D Line# 1      7                        Microsoft FORTRAN77 V3.13 8/05/83
               SUBROUTINE OUTAB(DMP,JFUN,LSA,LSJ,NRC,S)
          C
               INTEGER    JFUN(LSJ,1),NRC(LSA,2,1),NSA(4)
          C
               DIMENSION  DMP(LSA,1),S(LSA,1)
          C
               CHARACTER  FORMS(8)*21,TITL(72,2)*1
          C
               COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
              +               KRPM,NFGEN,NSA,NSL,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
              +               NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,IFLG1,
              +               THETA,SCALE,MAXPTS,IPCNT,ISCNT
          C
          C-----WRITE A CONNECTIVITY MATRIX FOR EACH INTERGROUP ADDITION
          C
               DO 100 J=1,NSL,3
1              WRITE(6,8002)
1              DO 90 I=J,J+2
2              WRITE(6,38) I,'ROT Y:',NRC(I*4-3,1,1),'ROT Z:',NRC(I*4-2,1,1),
2             +             'CAS Y:',NRC(I*4-3,2,2),'CAS Z:',NRC(I*4-2,2,2)
2         C
2              WRITE(6,22)
2         C
2              WRITE(6,24)
2              WRITE(6,26)
2              WRITE(6,40) S(I*4-3,1), S(I*4-1,1), DMP(I*4-3,1), DMP(I*4-1,1)
2              WRITE(6,26)
2              WRITE(6,42) JFUN(I*4-3,1),JFUN(I*4-1,1),
2             1            JFUN(I*4-3,2),JFUN(I*4-1,2)
2              WRITE(6,26)
2              WRITE(6,32)
2              WRITE(6,26)
2              WRITE(6,40) S(I*4,1), S(I*4-2,1), DMP(I*4,1), DMP(I*4-2,1)
2              WRITE(6,26)
2              WRITE(6,42) JFUN(I*4,1),JFUN(I*4-2,1),
2             1            JFUN(I*4,2),JFUN(I*4-2,2)
2              WRITE(6,26)
2              WRITE(6,24)
2           90 IF(I.EQ.NSL) GOTO 200
1          100 WRITE(6,8003)
          C
            22 FORMAT(12X,'INTERGROUP STIFFNESS',21X,'INTERGROUP DAMPING')
            24 FORMAT(2(5X,'***',27X,'***',2X))
            26 FORMAT(2(5X,'*',15X,':',15X,'*',2X))
            32 FORMAT(2(5X,'*',15('.'),':',15('.')'*',2X))
            38 FORMAT(//5X,'NSL NUMBER',I3,4(5X,A,I4)/)
            40 FORMAT(2(5X,'* ',1PE13.5,' : ',E13.5,' *',2X))
            42 FORMAT(2(5X,'*   FG (',I2,')',5X,':   FG (',I2,')',5X,'*',2X))
          8002 FORMAT(//5X,'* * * INTERGROUP ADDITIONS * * *')
          8003 FORMAT(1H1)
          C
          C-----WRITE INTRAGROUP ADDITIONS
          C
           200 IF(NSA(1)+NSA(2)+NSA(3)+NSA(4).NE.0) THEN
               WRITE(6,101)
               KK=0
               DO 300 II=1,4
1              IF(NSA(II).EQ.0)          GO TO 300
1              WRITE(6,8000) FORMS(II+4)
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.13 8/05/83
1              WRITE(6,8001)
1              DO 299  I  =   1,NSA(II)
2              N  =   4*NSL  +  I
2         299  WRITE(6,28) I,(NRC(N,J,II),J=1,2),S(N,II),JFUN(N+KK,1),DMP(N,II),
2            1                JFUN(N+KK,2)
1         300  KK=KK+NSA(II)
               ENDIF
        C
           28  FORMAT((5X,3(I3,4X),1PE12.5,4X,I3,4X,E12.5,4X,I3))
          101  FORMAT(1H1///,5X,'* * * GENERAL ADDITIONS * * *'//)
         8000  FORMAT(///10X,A,//)
         8001  FORMAT(5X,'NO.',4X,'ROW',3X,'COL.',4X,'STIFFNESS',5X,'FUNC.',4X,
             1           'DAMPING',7X,'FUNC.'/)
        C
               END
```

Name	Type	Offset	P	Class
DMP	REAL	0	*	
DRPM	REAL	188		/DATA /
FHIGH	REAL	194		/DATA /
FLOW	REAL	198		/DATA /
FORMS	CHAR*21	1164		
I	INTEGER*2	1340		
IFLG	INTEGER*2	204		/DATA /
IFLG1	INTEGER*2	206		/DATA /
II	INTEGER*2	1736		
IPCNT	INTEGER*2	218		/DATA /
IPLTF	INTEGER*2	172		/DATA /
IPRT	INTEGER*2	178		/DATA /
IPRT2	INTEGER*2	174		/DATA /
IPRT3	INTEGER*2	176		/DATA /
ISCNT	INTEGER*2	220		/DATA /
J	INTEGER*2	1332		
JFUN	INTEGER*2	4	*	
KCRT	INTEGER*2	180		/DATA /
KK	INTEGER*2	1734		
KRPM	INTEGER*2	158		/DATA /
LSA	INTEGER*2	8	*	
LSJ	INTEGER*2	12	*	
MAXPTS	INTEGER*2	216		/DATA /
N	INTEGER*2	1744		
NCASE	INTEGER*2	154		/DATA /
NCDOF	INTEGER*2	6		/DATA /
NCMOD	INTEGER*2	2		/DATA /
NFGEN	INTEGER*2	160		/DATA /
NGYRO	INTEGER*2	0		/DATA /
NI	INTEGER*2	192		/DATA /
NRC	INTEGER*2	16	*	
NRDOF	INTEGER*2	8		/DATA /
NRMOD	INTEGER*2	4		/DATA /
NROT	INTEGER*2	156		/DATA /
NSA	INTEGER*2	162		/DATA /
NSL	INTEGER*2	170		/DATA /
NSTAT	INTEGER*2	202		/DATA /
NXTC	INTEGER*2	182		/DATA /
RPM1	REAL	184		/DATA /
S	REAL	20	*	
SCALE	REAL	212		/DATA /

```
THETA  REAL           208   /DATA  /
TITL   CHAR*1          10   /DATA  /
```

```
*R3.FOR*****************************************************************
C
C          R3  READS AND PRINTS THE ROTOR AND CASING DEGREES OF
C              FREEDOM WHERE DISPLACEMENTS WILL BE CALCULATED.
C
C***********************************************************************
C
      SUBROUTINE R3(ZRPM,ICASE,IROT,X,NZ,RZ,RZ1)
C
      INTEGER     ICASE(1),IROT(1),NZ(4)
C
      DIMENSION   ZRPM(1),RZ(6),RZ1(6),X(10)
C
      CHARACTER   TITL(72,2)*1
C
      COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,NCASE,NROT,
     +                 KRPM,NFGEN,NSA(4),NSL,IPLTF,IPRT2,IPRT3,IPRT,KCRT,
     +                 NXTC,RPM1,DRPM,NI,FHIGH,FLOW,NSTAT,IFLG,IFLG1,
     +                 THETA,SCALE,MAXPTS,IPCNT,ISCNT
C
C-----READ DISPLACEMENT DOF DATA
C
      READ(5,200) NROT,NCASE
C
      IF(NROT.NE.0) THEN
        READ(5,210) (IROT(I),I=1,NROT)
        WRITE(6,220)
        WRITE(6,221) (IROT(I),I=1,NROT)
      ELSE
        WRITE(6,600)
      ENDIF
C
      IF(NCASE.NE.0) THEN
        READ(5,210) (ICASE(I),I=1,NCASE)
        WRITE(6,215)
        WRITE(6,221) (ICASE(I),I=1,NCASE)
      ELSE
        WRITE(6,610)
      ENDIF
C
C-----READ PRINT/PLOT FREQUENCY RANGE
C
      WRITE(6,320)
      READ(5,300) FHIGH,FLOW
      WRITE(6,310) FHIGH,FLOW
C
C-----READ CRITICAL SPEEDS AND STABILITY PLOTTING FLAGS AND DATA
C
      READ(5,12) RZ
      WRITE(6,61) RZ
C
      READ(5,101) IPLTF,IPRT2
      WRITE(6,51) IPLTF,IPRT2
C
      IF(IPRT2 .EQ. 0) THEN
```

```
          READ(5,12) RZ1
          WRITE(6,53) RZ1
          READ(5,210) NZ
          WRITE(6,54) NZ
        ELSE
          RZ1(3)=-200
          RZ1(4)= 100
        ENDIF
C
C-----READ MODE SHAPE PLOTTING DATA
C
        READ(5,101) IPRT3
        WRITE(6,63) IPRT3
C
        IF(IPRT3 .EQ. 0) THEN
          READ(5,210) KRPM
          WRITE(6,52) KRPM
          READ(5,12) (ZRPM(I),I=1,KRPM)
          WRITE(6,59) (ZRPM(I),I=1,KRPM)
          READ(5,92) NSTAT,IFLG,IFLG1,THETA,SCALE
          WRITE(6,58)
          WRITE(6,55) NSTAT,IFLG,IFLG1,THETA,SCALE
          READ(5,12) (X(I),I=1,NSTAT)
          WRITE(6,56)
          WRITE(6,57) (X(I),I=1,NSTAT)
        ENDIF
C
   12 FORMAT(BN,6F12.0)
   51 FORMAT(/10X,'IPLTF =',I3,19X,'IPRT2 =',I3)
   52 FORMAT(/10X,'MODE SHAPES PLOTTED FOR ',I2,' SPEED CASES AT',
     1        ' FOLLOWING SPEEDS')
   53 FORMAT(/10X,'XB1     =',1PE13.5,9X,'XR1     =',E13.5/10X,
     1        'YB1     =',E13.5,9X,'YT1     =',E13.5/10X,'DX1     =',
     2        E13.5,9X,'DY1     =',E13.5)
   54 FORMAT(/10X,'STABILITY PLOT CHARACTERS',2X,4(3X,I3))
   55 FORMAT(/10X,'STATIONS=',I3,4X,'IFLG=',I2,4X,'IFLG1=',
     1        I2,4X,'THETA=',F6.1,4X,'SCALE=',F5.3)
   56 FORMAT(/10X,'STATION LOCATIONS'/)
   57 FORMAT(15X,5F11.3)
   58 FORMAT(//10X,'MODE SHAPE PLOTTING DATA')
   59 FORMAT(/15X,5F11.0)
   61 FORMAT(/10X,'XB     =',1PE13.5,9X,'XR     =',E13.5/10X,'YB     =',
     1        E13.5,9X,'YT     =',E13.5/10X,'DX     =',E13.5,9X,
     2        'DY     =',E13.5)
   63 FORMAT(/10X,'IPRT3 =',I3)
   92 FORMAT(BN,I6,2I3,2F12.6)
  101 FORMAT(BN,I2)
  200 FORMAT(BN,2I6)
  210 FORMAT(BN,24I3)
  215 FORMAT(6(/),5X,'CASING DISPLACEMENTS WILL BE COMPUTED',
     1        ' AT THE FOLLOWING DEGREES OF FREEDOM'//)
  220 FORMAT(1H1///5X,'ROTOR DISPLACEMENTS WILL BE COMPUTED',
     1        ' AT THE FOLLOWING DEGREES OF FREEDOM'//)
  221 FORMAT(10X,12I5)
  300 FORMAT(BN,2E12.5)
  310 FORMAT(///10X,'FHIGH =',1PE13.5,' CPM',5X,'FLOW =',E13.5,' CPM')
  320 FORMAT(1H1///5X,'* *  PRINT AND PLOT CONTROL OPTIONS  * *')
  600 FORMAT(1H1///5X,'ROTOR DISPLACEMENTS WILL NOT BE COMPUTED')
  610 FORMAT(6(/),5X,'CASING DISPLACEMENTS WILL NOT BE COMPUTED')
```

```
D Line# 1      7
    446 C
    447         END
```

Name	Type	Offset	P	Class	
DRPM	REAL	188		/DATA	/
FHIGH	REAL	194		/DATA	/
FLOW	REAL	198		/DATA	/
I	INTEGER*2	1980			
ICASE	INTEGER*2	4	*		
IFLG	INTEGER*2	204		/DATA	/
IFLG1	INTEGER*2	206		/DATA	/
IPCNT	INTEGER*2	218		/DATA	/
IPLTF	INTEGER*2	172		/DATA	/
IPRT	INTEGER*2	178		/DATA	/
IPRT2	INTEGER*2	174		/DATA	/
IPRT3	INTEGER*2	176		/DATA	/
IROT	INTEGER*2	8	*		
ISCNT	INTEGER*2	220		/DATA	/
KCRT	INTEGER*2	180		/DATA	/
KRPM	INTEGER*2	158		/DATA	/
MAXPTS	INTEGER*2	216		/DATA	/
NCASE	INTEGER*2	154		/DATA	/
NCDOF	INTEGER*2	6		/DATA	/
NCMOD	INTEGER*2	2		/DATA	/
NFGEN	INTEGER*2	160		/DATA	/
NGYRO	INTEGER*2	0		/DATA	/
NI	INTEGER*2	192		/DATA	/
NRDOF	INTEGER*2	8		/DATA	/
NRMOD	INTEGER*2	4		/DATA	/
NROT	INTEGER*2	156		/DATA	/
NSA	INTEGER*2	162		/DATA	/
NSL	INTEGER*2	170		/DATA	/
NSTAT	INTEGER*2	202		/DATA	/
NXTC	INTEGER*2	182		/DATA	/
NZ	INTEGER*2	16	*		
RPM1	REAL	184		/DATA	/
RZ	REAL	20	*		
RZ1	REAL	24	*		
SCALE	REAL	212		/DATA	/
THETA	REAL	208		/DATA	/
TITL	CHAR*1	10		/DATA	/
X	REAL	12	*		
ZRPM	REAL	0	*		

Name	Type	Size	Class
ADDM			SUBROUTINE
ADDS			SUBROUTINE
DATA		222	COMMON
FORT02			SUBROUTINE
OUTAB			SUBROUTINE
R3			SUBROUTINE

```
 Pass One     No Errors Detected
             447 Source Lines
```

```
*RSTAB*****************************************************************
C
C     P R O G R A M   R S T A B
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C**********************************************************************
C
C     PRINCIPAL SUBROUTINES UTILIZED IN RSTAB ARE CALLED
C     IN THE FOLLOWING ORDER:
C
C     PLOT . . . . . . . . . . . . . . . SET UP ROOT LOCUS PLOT
C
C     FORT03 . . . . . . . . . . . . . . SOLVE FOR EIGENVALUES VS. RPM
C
C              MATX
C              EIGRF
C              ANSR
C
C**********************************************************************
C
C     INPUT FILE NAMES    DESCRIPTION
C     ----------------    ----------------------------------------------
C
C     RUNDATA.BIN         BINARY RUN DATA FILE CREATED BY PROGRAM
C                         PRERSTAB
C
C     FIN.BAT             FORMATTED IBM BATCH FILE USED TO DETERMINE
C                         WHETHER TO PRINT THE OUTPUT LISTING DATA
C                         IMMEDIATELY, OR TO WRITE THE INFORMATION TO
C                         TEMPORARY FILE 'LIST2'
C
C     OUTPUT FILE NAMES   DESCRIPTION
C     -----------------   ----------------------------------------------
C
C     EIGENS.BIN          BINARY DATA FILE CONTAINING THE EIGENVALUES
C                         FOR EACH RPM STEP. USED BY PROGRAM PSTRSTAB
C                         FOR CRITICAL SPEED AND STABILITY PLOTTING
C
C     SHAPES.BIN          BINARY DATA FILE CONTAINING THE COMPLEX MODE
C                         SHAPES (IN NORMAL COORDINATES) FOR PLOTTING BY
C                         PROGRAM PSTRSTAB
C
C     LIST2               FORMATTED OUTPUT LISTING CREATED ONLY IF
C                         OUTPUT IS NOT PRINTED IMMEDIATELY
C
C*********************************************************************
$STORAGE:2
C
      PROGRAM RSTAB
C
      CHARACTER    TITL(72,2)*1,TAPE4*14
C
      INTEGER      NZ(4),NSA(4)
C
      REAL         RZ(6),RZ1(6)
C
      EQUIVALENCE (G,Z) , (TITL,TAPE4)
C
```

```
C* ARRAY DIMENSIONS MOST LIKELY TO CHANGE ****************************
C
C-----NOTES: DIMENSION WK TO (MAXO(LRDOF,LCDOF,2*LDYN))
C                              FUNC TO (NFGEN + 1)
C                              Z TO (2,LDYN,LDYN)
C                              W TO (2,LDYN)
C
      DIMENSION      AC(50,14),AR(115,16),WC(14),WR(16),ZETAC(14),X(10),
     +               SPEED(20,24),DMP(100,4),S(100,4),FG(20,24),
     +               G(115,115),FUNC(25),ZRPM(25),W(2,60)
C
      INTEGER        JFUN(160,2),NPT(24),NRC(100,2,4),IROT(20),ICASE(20)
C
      REAL*8         GBAR(16,16),A(60,60),WK(120),Z(2,60,60)
C
C*********************************************************************
C
      COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
     +                 NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
     +                 IPLTF,IPRT2,IPRT3
      COMMON     /MEM/Z
      COMMON     /MOD/AR,AC
C
C-----DEFAULT UNITY FUNCTION GENERATOR
C
      DATA FUNC(1)/1.0/
C
C* DATA STATEMENTS NEEDED FOR ARRAY REDIMENSIONING ********************
C
      DATA LCMOD,LRMOD/14,16/,          LSA,LSJ      /100,160/
      DATA LPT         /20/,            LRDOF,LCDOF/115,50/
C
C*********************************************************************
C
      LDYN=2*(LRMOD+LCMOD)
      LG=MAXO(LRDOF,LCDOF)
C
C-----OPEN OUTPUT FILE
C     NOTES: FILE FIN.BAT IS EMPTY IF OUTPUT IS BEING LISTED ON PRN,
C            ELSE TAPE4='LIST2'
C
C            CHANNEL 4 IS USED FOR PRINTING THE ROOT-LOCUS PLOT.
C
      OPEN(2,FILE='FIN.BAT',STATUS='OLD')
      TAPE4='PRN'
      READ(2,9000,END=5) TAPE4
    5 CLOSE(2)
      OPEN(6,FILE=TAPE4,STATUS='NEW')
      OPEN(4,FILE='PRN',STATUS='NEW')
C
C-----OPEN RUN DATA, EIGENVALUE, AND MODE SHAPE DATA FILES
C
      OPEN(1,FILE='RUNDATA.BIN',STATUS='OLD',FORM='UNFORMATTED')
      OPEN(2,FILE='EIGENS.BIN',STATUS='NEW',FORM='UNFORMATTED')
      OPEN(3,FILE='SHAPES.BIN',STATUS='NEW',FORM='UNFORMATTED')
C
C-----READ PROBLEM DATA
C
   10 READ(1) (TITL(I,1),I=1,72),NCDOF,NCMOD,NRDOF,NRMOD,IPCNT
```

```
      C
            READ(1)  ((AC(I,J),I=1,NCDOF),J=1,NCMOD),((AR(I,J),I=1,NRDOF),
           +          J=1,NRMOD),((GBAR(I,J),I=1,NRMOD),J=1,NRMOD),(WC(I),
           +          I=1,NCMOD),(WR(I),I=1,NRMOD)
      C
      C-----READ SUBCASE DATA
      C
         20 READ(1)  (TITL(I,2),I=1,72),NSA,NSL,IPRT,KCRT,NXTC,NFGEN,MAXPTS,
           +          RPM1,DRPM,NI,NROT,NCASE,FHIGH,FLOW,RZ,IPLTF,IPRT2,RZ1,
           +          NZ,IPRT3,KRPM,NSTAT,IFLG,IFLG1,THETA,SCALE,ISCNT
      C
            NSJ=4*NSL+NSA(1)+NSA(2)+NSA(3)+NSA(4)
            NNSA=4*NSL+MAX0(NSA(1),NSA(2),NSA(3),NSA(4))
      C
            READ(1)  (ZETAC(I),I=1,NCMOD),(NPT(J),(FG(I,J),SPEED(I,J),
           +          I=1,MAXPTS),J=1,NFGEN),((JFUN(I,J),I=1,NSJ),J=1,2),
           +          ((S(I,J),DMP(I,J),(NRC(I,K,J),K=1,2),I=1,NNSA),J=1,4),
           +          (IROT(I),I=1,NROT),(ICASE(I),I=1,NCASE),(ZRPM(I),
           +          I=1,KRPM),(X(I),I=1,NSTAT)
      C
      C-----SAVE CRITICAL SPEED AND STABILITY PLOT DATA
      C
            WRITE(2)  IPLTF,IPRT2,TITL,RZ,RZ1,RPM1,DRPM,NI,FHIGH,FLOW,
           +           NRMOD,NCMOD,NZ,KCRT,IPCNT,ISCNT,NXTC
      C
      C-----SAVE MODE SHAPE PLOTTING DATA
      C
            WRITE(3)  IPRT,IPRT3,NCDOF,NCMOD,NRDOF,NRMOD,NCASE,NROT,TITL,
           +           NSTAT,IFLG,IFLG1,THETA,SCALE,IPCNT,ISCNT,NXTC
      C
            WRITE(3)  ((AC(I,J),I=1,NCDOF),J=1,NCMOD),((AR(I,J),I=1,NRDOF),
           +           J=1,NRMOD),(ICASE(I),I=1,NCASE),(IROT(I),I=1,NROT),
           +           (X(I),I=1,NSTAT)
      C
      C-----SET UP PLOTTING GRID
      C
            CALL PLOT(RZ,RZ1,RPM1)
      C
      C-----F O R T O 3
      C
            WRITE(6,9002) IPCNT,ISCNT
            CALL FORT03(A,AC,AR,DMP,FG,GBAR,JFUN,NPT,NRC,W,S,SPEED,WC,WR,
           +               ZETAC,FUNC(2),LCDOF,LRDOF,LRMOD,LDYN,LSA,LSJ,LPT,
           +               LG,ICASE,IROT,WK,G,Z,ZRPM)
      C
      C-----MAKE HARDCOPY OF THE ROOT LOCUS DISPLAY
      C
            WRITE(4,9001)
            CALL QPSCR
      C
      C-----RERUN?
      C
            GOTO(10,20,20) NXTC
      C
      C-----CLOSE DATA FILES
      C
            CLOSE(1)
            CLOSE(2)
            CLOSE(3)
```

```
C
C-----TERMINATE PLOTTING MODE
C
      CALL QSMODE(3)
      CALL QCSIZ(0,0)
      CALL QCLEAR(1,7)
      CALL QBORD(1)
C
C-----FORMAT STATEMENTS
C
 9000 FORMAT(/4X,A)
 9001 FORMAT(1H1/////)
 9002 FORMAT(1H1///10X,'* * *   M A I N   R O U T I N E   * * *'//15X,
     +        'MODAL INPUT ',I2,5X,'SUBCASE ',I2)
C
      END
```

Name	Type	Offset	P	Class
A	REAL*8	13370		
AC	REAL	7360		/MOD /
AR	REAL	0		/MOD /
DMP	REAL	9750		
DRPM	REAL	178		/DATA /
FG	REAL	11350		
FHIGH	REAL	184		/DATA /
FLOW	REAL	188		/DATA /
FUNC	REAL	13270		
G	REAL	0		/MEM /
GBAR	REAL*8	7702		
I	INTEGER*2	42188		
ICASE	INTEGER*2	7662		
IFLG	INTEGER*2	42218		
IFLG1	INTEGER*2	42220		
IPCNT	INTEGER*2	42192		
IPLTF	INTEGER*2	192		/DATA /
IPRT	INTEGER*2	168		/DATA /
IPRT2	INTEGER*2	194		/DATA /
IPRT3	INTEGER*2	196		/DATA /
IROT	INTEGER*2	7622		
ISCNT	INTEGER*2	42230		
J	INTEGER*2	42194		
JFUN	INTEGER*2	5334		
K	INTEGER*2	42236		
KCRT	INTEGER*2	170		/DATA /
KRPM	INTEGER*2	154		/DATA /
LCDOF	INTEGER*2	42182		
LCMOD	INTEGER*2	42170		
LDYN	INTEGER*2	42184		
LG	INTEGER*2	42186		
LPT	INTEGER*2	42178		
LRDOF	INTEGER*2	42180		
LRMOD	INTEGER*2	42172		
LSA	INTEGER*2	42174		
LSJ	INTEGER*2	42176		
MAX0				INTRINSIC
MAXPTS	INTEGER*2	42210		
NCASE	INTEGER*2	42214		
NCDOF	INTEGER*2	6		/DATA /

```
NCMOD   INTEGER*2         2   /DATA   /
NFGEN   INTEGER*2       156   /DATA   /
NGYRO   INTEGER*2         0   /DATA   /
NI      INTEGER*2       182   /DATA   /
NNSA    INTEGER*2     42234
NPT     INTEGER*2      5974
NRC     INTEGER*2      6022
NRDOF   INTEGER*2         8   /DATA   /
NRMOD   INTEGER*2         4   /DATA   /
NROT    INTEGER*2     42212
NSA     INTEGER*2       158   /DATA   /
NSJ     INTEGER*2     42232
NSL     INTEGER*2       166   /DATA   /
NSTAT   INTEGER*2     42216
NXTC    INTEGER*2       172   /DATA   /
NZ      INTEGER*2      1758
RPM1    REAL            174   /DATA   /
RZ      REAL           1766
RZ1     REAL           1790
S       REAL           3734
SCALE   REAL          42226
SPEED   REAL           1814
TAPE4   CHAR*14          10   /DATA   /
THETA   REAL          42222
TITL    CHAR*1           10   /DATA   /
W       REAL           1278
WC      REAL            158
WK      REAL*8          318
WR      REAL            214
X       REAL            278
Z       REAL*8            0   /MEM    /
ZETAC   REAL              2
ZRPM    REAL             58
```

```
*PLOT.FOR***************************************************************
C
C      S U B R O U T I N E   P L O T
C
C      ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C***********************************************************************
C
       SUBROUTINE PLOT(RZ,RZ1,RPM1)
C
       CHARACTER     NUMBER*6,NUM2(10)*1
C
       EQUIVALENCE  (NUM2(5),NUMBER)
C
       DIMENSION     RZ(6),RZ1(6),ZETA(4)
C
       EXTERNAL      JCOL,JROW
C
C-----COMMON
C
       COMMON /QQYTIC/KNTY,JYTICC(30),JYTICR(30)
       COMMON /RANGE/XRANGE(5),YRANGE(5)
C
C-----ZETA = SQRT(1-CR*CR)*9.54929 / CR ;
```

```
C       WHERE: CR=CRITICAL DAMPING RATIO
C
        DATA ZETA/954.9,381.8,190.7,95.01/
        DATA NUM2/'R','P','M','=',6*' '/
        DATA I1,I2,I3,I4,I5,I8/1,2,3,4,5,8/
C
C-----INITIALIZE CRT PLOT
C
        CALL QSMODE(6)
C
C-----SCALE AXES
C
        XRANGE(I1)  =  RZ1(I3)
        XRANGE(I2)  =  RZ1(I4)
      5 CALL SCALE(XRANGE,8.,I2,I1)
        XRANGE(I5)  =  XRANGE(I3)  +  I8 * XRANGE(I4)
C.....ENSURE ENOUGH ROOM FOR TITLING
        IF(XRANGE(I5)/XRANGE(I4).LT.1.9) THEN
          XRANGE(I2)=XRANGE(I2)+XRANGE(I4)
          GOTO 5
        ELSEIF(XRANGE(I3)/XRANGE(I4).GT.-1.9) THEN
          XRANGE(I1)=XRANGE(I1)-XRANGE(I4)
          GOTO 5
        ENDIF
C
        YRANGE(I1)  =  0.
        YRANGE(I2)  =  RZ(I4)
        CALL SCALE(YRANGE,5.,I2,I1)
        YRANGE(I5)  =  YRANGE(I3)  +  I5 * YRANGE(I4)
C
C-----DEFINE PLOT PIXEL WINDOW
C
        CALL QPLOT(I1,638,I1,198,XRANGE(I3),XRANGE(I5),0.,YRANGE(I5),
     +            0.,0.,0,1.,1.)
C
C-----DRAW AXES
C
        CALL QAXES(XRANGE(I3),XRANGE(I5),YRANGE(I3),YRANGE(I5),XRANGE(I4),
     +            XRANGE(I4)/I2,YRANGE(I4),YRANGE(I4)/I2)
C
C-----LABEL AXES
C
        CALL QPTXT(I4,'REAL',I3,74,I1)
        CALL QPTXT(10,'IMAG. (CPM)',I3,JYTICC(KNTY)/I8+I3,23)
C
        WRITE(NUMBER,8000) XRANGE(I4)
        CALL QPTXT(6,NUMBER,I3,JCOL(XRANGE(I4))/I8-I3,I1)
C
        WRITE(NUMBER,8000) YRANGE(I4)
        CALL QPTXT(6,NUMBER,I3,JYTICC(KNTY)/I8+I3,JROW(YRANGE(I4))/I8)
C
C-----DRAW BORDER
C
        CALL QLINE(0,I1,0,198,I3)
        CALL QRAST(I1,198,I3,636)
        CALL QLINE(637,197,637,I1,I3)
C
C-----DRAW DAMPING INTERVALS
C
```

```
D Line# 1      7
               JC=JCOL(-YRANGE(I5)/ZETA(I1))
               IF(JC.GT.80) THEN
                 CALL QLINE(JC,197,JC,196,I3)
                 CALL QPTXT(I2,'1%',I3,JC/I8,23)
               ENDIF
         C
               JC=JCOL(-YRANGE(I5)/ZETA(I2))
               IF(JC.GT.80) THEN
                 CALL QLINE(JC,197,JC,196,I3)
                 CALL QPTXT(I4,'2.5%',I3,JC/I8-I1,23)
               ENDIF
         C
               JR=JROW(-XRANGE(I3)*ZETA(I3))
               IF(JR.LT.174) THEN
                 CALL QRAST(I1,JR,I3,I3)
                 CALL QPTXT(I2,'5%',I3,1,JR/I8+I1)
               ENDIF
         C
               JR=JROW(-XRANGE(I3)*ZETA(I4))
               IF(JR.LT.174) THEN
                 CALL QRAST(I1,JR,I3,I3)
                 CALL QPTXT(I3,'10%',I3,1,JR/I8+I1)
               ENDIF
         C
               CALL QPTXT(7,'DAMPING',I3,I1,23)
         C
         C-----WRITE STARTING RPM
         C
               WRITE(NUMBER,8000) RPM1
               CALL QPTXT(10,NUM2,I3,69,22)
         C
         C-----FORMAT STATEMENTS
         C
          8000 FORMAT(F6.0)
         C
               END
```

Name	Type	Offset	P	Class
I1	INTEGER*2	42420		
I2	INTEGER*2	42422		
I3	INTEGER*2	42424		
I4	INTEGER*2	42426		
I5	INTEGER*2	42428		
I8	INTEGER*2	42430		
JC	INTEGER*2	42432		
JCOL				EXTERNAL
JR	INTEGER*2	42434		
JROW				EXTERNAL
JYTICC	INTEGER*2	2		/QQYTIC/
JYTICR	INTEGER*2	62		/QQYTIC/
KNTY	INTEGER*2	0		/QQYTIC/
NUM2	CHAR*1	42410		
NUMBER	CHAR*6	42414		
RPM1	REAL	8	*	
RZ	REAL	0	*	
RZ1	REAL	4	*	
XRANGE	REAL	0		/RANGE /
YRANGE	REAL	20		/RANGE /

ZETA	REAL	42394		

```
*JCOL**************************************************************
C
C      F U N C T I O N   J C O L
C
C      ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C*******************************************************************
C
       FUNCTION JCOL(X)
C
       COMMON  /RANGE/XRANGE(5),YRANGE(5)
C
       JCOL=637*(X-XRANGE(3))/(XRANGE(5)-XRANGE(3))+1
C
       END
```

Name	Type	Offset	P	Class
X	REAL	0	*	
XRANGE	REAL	0		/RANGE /
YRANGE	REAL	20		/RANGE /

```
*JROW**************************************************************
C
C      F U N C T I O N   J R O W
C
C      ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C*******************************************************************
C
       FUNCTION JROW(Y)
C
       COMMON  /RANGE/XRANGE(5),YRANGE(5)
C
       JROW=197*(Y-YRANGE(3))/(YRANGE(5)-YRANGE(3))+1
C
       END
```

Name	Type	Offset	P	Class
XRANGE	REAL	0		/RANGE /
Y	REAL	0	*	
YRANGE	REAL	20		/RANGE /

Name	Type	Size	Class
DATA		198	COMMON
FORT03			SUBROUTINE
JCOL	INTEGER*2		FUNCTION
JROW	INTEGER*2		FUNCTION
MEM		57600	COMMON
MOD		10160	COMMON
PLOT			SUBROUTINE

```
C Line# 1      7                                     Microsoft FORTRAN77 V3.13 8/05/83
QAXES                    SUBROUTINE
QBOFD                    SUBROUTINE
QCLEAR                   SUBROUTINE
QCSII                    SUBROUTINE
QLINE                    SUBROUTINE
QPLOT                    SUBROUTINE
QPSIR                    SUBROUTINE
QPTAT                    SUBROUTINE
QOPTIC             122   COMMON
QRAST                    SUBROUTINE
QSMODE                   SUBROUTINE
RANGE               40   COMMON
RSTAB                    PROGRAM
SCALE                    SUBROUTINE

 Pass One     No Errors Detected
             342 Source Lines
```

```
D Line# 1      7                       Microsoft FORTRAN77 V3.13 8/05/83
  *FORT03.FOR*************************************************************
  C
  C      S U B R O U T I N E   F O R T 0 3
  C
  C***********************************************************************
  C
  C      SUBROUTINES UTILIZED IN FORT03 ARE CALLED
  C      IN THE FOLLOWING ORDER:
  C
  C      MATX . . . . . . . . . . . FORMS THE MATRIX ADDITIONS INTO THE
  C                                 NORMALIZED DYNAMICAL MATRIX "A"
  C
  C      EIGRF. . . . . . . . . . . FROM THE IMSL LIBRARY. SOLVES A GENERAL
  C                                 MATRIX FOR COMPLEX ROOTS AND EIGENVECTORS
  C
  C      ANSR . . . . . . . . . . . SAVES ITERATION STEP RESULTS AND PLOTS
  C                                 EIGENVALUES ON CRT (ROOT LOCUS)
  C
  C***********************************************************************
  C
  $STORAGE:2
  C
         SUBROUTINE FORT03(A,AC,AR,DMP,FG,GBAR,JFUN,NPT,NRC,W,S,SPEED,WC,
       +                   WR,ZETAC,FUNC,LCDOF,LRDOF,LRMOD,LDYN,LSA,LSJ,
       +                   LPT,LG,ICASE,IROT,WK,G,Z,ZRPM)
  C
         CHARACTER TITL(72,2)*1
  C
         REAL*8    A(LDYN,1),WK(1),GBAR(LRMOD,1),Z(2,LDYN,1)
  C
         INTEGER   JFUN(LSJ,1),NPT(1),NRC(LSA,2,1),ICASE(1),IROT(1),NZ(4),
       +           NSA(4)
  C
         DIMENSION AC(LCDOF,1),AR(LRDOF,1),WC(1),WR(1),S(LSA,1),G(LG,1),
       +           DMP(LSA,1),FG(LPT,1),SPEED(LPT,1),ZETAC(1),FUNC(1),
       +           ZRPM(1),W(2,1)
  C
         COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
       +                 NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
       +                 IPLTF,IPRT2,IPRT3
  C
  C-----M A I N   R O U T I N E
  C
         NMODES=NRMOD+NCMOD
         DO 200 IRPM = 1,NI+1
1          RPM  =  RPM1 + (IRPM - 1) * DRPM
1          WRITE(6,36) TITL,RPM
1 C
1 C-------UPDATE FUNCTION GENERATORS
1 C
1          DO 100 I=1,NFGEN
2          DO 10 J=1,NPT(I)-1
3     10   IF(RPM.LT.SPEED(J,I)) GOTO 100
2    100   FUNC(I)=FG(J-1,I)+(FG(J,I)-FG(J-1,I))*(RPM-SPEED(J-1,I))/
2        +         (SPEED(J,I)-SPEED(J-1,I))
1 C
1 C-------ZERO THE DYNAMICAL MATRIX
1 C
1          DO 400 I=1,2*NMODES
```

```
D Line# 1      7
2                DO 400 J=1,2*NMODES
3         400   A(I,J)=0.0
1      C
1      C-------SET THE IDENTITY MASS PARTITION OF A
1      C
1               DO 300 I = 1,NMODES
2         300   A(I+NMODES,I) = 1.0
1      C
1      C-------LOAD NSL'S AND NSA'S INTO A
1      C
1               CALL MATX(AC,AR,DMP,JFUN,LCDOF,LRDOF,LSA,LSJ,LDYN,NRC,S,
1            +                 A,G,LG,WK,FUNC)
1      C
1      C-------ADD SPECTRAL MATRIX, DAMPING MATRIX, AND GYROSCOPICS TO A
1      C
1               DO 410      I = 1, NRMOD
2                 A(I,NMODES+I) = A(I,NMODES+I) - WR(I)*WR(I)
2                 DO 410      J = 1, NRMOD
3         410   A(I,J) = A(I,J) - GBAR(I,J)*RPM*0.1047197
1      C
1               DO 430      I = 1, NCMOD
2                 IR   =   NRMOD  +  I
2                 A(IR,IR)   =   A(IR,IR)   -   2.*ZETAC(I)*WC(I)
2         430   A(IR,NMODES+IR) = A(IR,NMODES+IR) - WC(I)*WC(I)
1      C
1      C-------CALCULATE EIGENVALUES AND EIGENVECTORS
1      C
1               CALL EIGRF(A,2*NMODES,LDYN,W,Z,LDYN,WK,IER)
1      C
1      C-------STORE RESULTS
1      C
1               CALL ANSR(Z,LDYN,RPM,W,ZRPM)
1      C
1         200 CONTINUE
       C
       C-----WRITE END-OF-CASE RECORD ON EIGENVECTOR FILE
       C
             WRITE(3) RPM,-9999,(W(J,1),J=1,2),
            +                      ((SNGL(Z(J,L,1)),J=1,2),L=1,NMODES)
       C
       C-----FORMAT STATEMENTS
       C
          36 FORMAT(1H1/5X,72A//5X,72A///5X,'ROTOR SPIN SPEED =',F8.0,'   RPM')
       C
             END
```

Name	Type	Offset	P	Class
A	REAL*8	0	*	
AC	REAL	4	*	
AR	REAL	8	*	
DMP	REAL	12	*	
DRPM	REAL	178		/DATA /
FG	REAL	16	*	
FHIGH	REAL	184		/DATA /
FLOW	REAL	188		/DATA /
FUNC	REAL	60	*	
G	REAL	108	*	
GBAR	REAL*8	20	*	

```
I       INTEGER*2      24
ICASE   INTEGER*2      96 *
IER     INTEGER*2      78
IPLTF   INTEGER*2     192    /DATA   /
IPRT    INTEGER*2     168    /DATA   /
IPRT2   INTEGER*2     194    /DATA   /
IPRT3   INTEGER*2     196    /DATA   /
IR      INTEGER*2      76
IROT    INTEGER*2     100 *
IRPM    INTEGER*2      12
J       INTEGER*2      32
JFUN    INTEGER*2      24 *
KCRT    INTEGER*2     170    /DATA   /
KRPM    INTEGER*2     154    /DATA   /
L       INTEGER*2      82
LCDOF   INTEGER*2      64 *
LDYN    INTEGER*2      76 *
LG      INTEGER*2      92 *
LPT     INTEGER*2      88 *
LRDOF   INTEGER*2      68 *
LRMOD   INTEGER*2      72 *
LSA     INTEGER*2      80 *
LSJ     INTEGER*2      84 *
NCDOF   INTEGER*2       6    /DATA   /
NCMOD   INTEGER*2       2    /DATA   /
NFGEN   INTEGER*2     156    /DATA   /
NGYRO   INTEGER*2       0    /DATA   /
NI      INTEGER*2     182    /DATA   /
NMODES  INTEGER*2      10
NPT     INTEGER*2      28 *
NRC     INTEGER*2      32 *
NRDOF   INTEGER*2       8    /DATA   /
NRMOD   INTEGER*2       4    /DATA   /
NSA     INTEGER*2     158    /DATA   /
NSL     INTEGER*2     166    /DATA   /
NXTC    INTEGER*2     172    /DATA   /
NZ      INTEGER*2       2
RPM     REAL           20
RPM1    REAL          174    /DATA   /
S       REAL           40 *
SNGL                         INTRINSIC
SPEED   REAL           44 *
TITL    CHAR*1         10    /DATA   /
W       REAL           36 *
WC      REAL           48 *
WK      REAL*8        104 *
WR      REAL           52 *
Z       REAL*8        112 *
ZETAC   REAL           56 *
ZRPM    REAL          116 *
```

```
  105 *MATX.FOR*****************************************************************
  106 C
  107 C          MATX  COORDINATES THE INTRAGROUP, INTERGROUP, MATRIX
  108 C                ADDITIONS AND THE CALCULATION OF THEIR STIFFNESS
  109 C                AND DAMPING VALUES (MULTIPLIED BY THE RESPECTIVE
  110 C                FUNCTION GENERATOR).
  111 C
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.13 8/05/83
     C**********************************************************************
     C
           SUBROUTINE MATX(AC,AR,DMP,JFUN,LCDOF,LRDOF,LSA,LSJ,LDYN,NRC,S,
          +                  A,G,LG,WK,FUNC)
     C
           INTEGER    JFUN(LSJ,1),NRC(LSA,2,1),NSA(4)
     C
           DIMENSION  AR(LRDOF,1),AC(LCDOF,1),DMP(LSA,1),S(LSA,1),
          +           G(LG,1),FUNC(1)
     C
           REAL*8     A(LDYN,1),WK(1)
     C
           CHARACTER TITL(72,2)*1
     C
           COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
          +                NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
          +                IPLTF,IPRT2,IPRT3
     C
     C-----ZERO THE G MATRIX
     C
           DO 100     I = 1,LG
1          DO 100     J = 1,LG
2      100 G(J,I)  = 0.0
     C
     C-----TRANSFORM MATRIX ADDITIONS INTO NORMAL COORDINATES,
     C     AND ADD THEM TO THE SYSTEM DYNAMICAL MATRIX "A".
     C
     C     1) ROTOR-TO-ROTOR ADDITIONS
     C     2) ROTOR-TO-CASING ADDITIONS
     C     3) CASING-TO-ROTOR ADDITIONS
     C     4) CASING-TO-CASING ADDITIONS
     C
           N   =  4*NSL
           NMODES = NRMOD + NCMOD
           IK=0
     C
           DO 1000 IP = 1,4
1    C
1            IF  ( NSL+NSA(IP) .EQ. 0 ) GOTO 1000
1    C
1    C-------ADD STIFFNESS ADDITIONS
1    C
1            IF(NSL.NE.0) THEN
1              DO 200 I = 1,N
2      200     G(NRC(I,1,IP),NRC(I,2,IP))  =  S(I,IP)*FUNC(JFUN(I,1))  +
2         1                                   G(NRC(I,1,IP),NRC(I,2,IP))
1            ENDIF
1    C
1            IF(NSA(IP).NE.0) THEN
1              DO 250 I   =  1,NSA(IP)
2              NN=N+I
2      250     G(NRC(NN,1,IP),NRC(NN,2,IP))=S(NN,IP)*FUNC(JFUN(NN+IK,1))+
2         1                                 G(NRC(NN,1,IP),NRC(NN,2,IP))
1            ENDIF
1    C
1            IF(IP.EQ.1) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AR,LRDOF,
1         1                 NRDOF,NRMOD,A(1,NMODES+1),LDYN,WK)
1            IF(IP.EQ.2) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AC,LCDOF,
1         1                 NCDOF,NCMOD,A(1,NMODES+NRMOD+1),LDYN,WK)
```

```
D Line# 1      7                                Microsoft FORTRAN77 V3.13 8/05/83
1              IF(IP.EQ.3) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AR,LRDOF,
1          1                    NRDOF,NRMOD,A(NRMOD+1,NMODES+1),LDYN,WK)
1              IF(IP.EQ.4) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AC,LCDOF,
1          1                    NCDOF,NCMOD,A(NRMOD+1,NMODES+NRMOD+1),LDYN,WK)
1        C
1        C-------RESET G
1        C
1              DO 300     I = 1, NSA(IP)+N
2        300   G(NRC(I,1,IP),NRC(I,2,IP)) = 0.0
1        C
1        C-------ADD DAMPING MODIFICATIONS
1        C
1              IF(NSL.NE.0) THEN
1                DO 350  I  =  1,N
2        350     G(NRC(I,1,IP),NRC(I,2,IP))  =  DMP(I,IP)*FUNC(JFUN(I,2))  +
2          1                                    G(NRC(I,1,IP),NRC(I,2,IP))
1              ENDIF
1        C
1              IF(NSA(IP).NE.0) THEN
1                DO 400 I  =  1,NSA(IP)
2                NN=N+I
2        400     G(NRC(NN,1,IP),NRC(NN,2,IP))=DMP(NN,IP)*FUNC(JFUN(NN+IK,2))+
2          1                                  G(NRC(NN,1,IP),NRC(NN,2,IP))
1              ENDIF
1        C
1              IF(IP.EQ.1) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AR,LRDOF,
1          1                         NRDOF,NRMOD,A,LDYN,WK)
1              IF(IP.EQ.2) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AC,LCDOF,
1          1                        NCDOF,NCMOD,A(1,NRMOD+1),LDYN,WK)
1              IF(IP.EQ.3) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AR,LRDOF,
1          1                         NRDOF,NRMOD,A(NRMOD+1,1),LDYN,WK)
1              IF(IP.EQ.4) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AC,LCDOF,
1          1                         NCDOF,NCMOD,A(NRMOD+1,NRMOD+1),LDYN,WK)
1        C
1        C-------RESET G
1        C
1              DO 450     I = 1, NSA(IP)+N
2        450   G(NRC(I,1,IP),NRC(I,2,IP)) = 0.0
1        C
1       1000 IK = IK + NSA(IP)
         C
             END
```

```
Name    Type         Offset P Class

A       REAL*8          44 *
AC      REAL             0 *
AR      REAL             4 *
DMP     REAL             8 *
DRPM    REAL           178   /DATA  /
FHIGH   REAL           184   /DATA  /
FLOW    REAL           188   /DATA  /
FUNC    REAL            60 *
G       REAL            48 *
I       INTEGER*2      148
IK      INTEGER*2      168
IP      INTEGER*2      170
IPLTF   INTEGER*2      192   /DATA  /
IPRT    INTEGER*2      168   /DATA  /
```

```
IPRT2   INTEGER*2      194   /DATA   /
IPRT3   INTEGER*2      196   /DATA   /
J       INTEGER*2      156
JFUN    INTEGER*2       12 *
KCRT    INTEGER*2      170   /DATA   /
KRPM    INTEGER*2      154   /DATA   /
LCDOF   INTEGER*2       16 *
LDYN    INTEGER*2       32 *
LG      INTEGER*2       52 *
LRDOF   INTEGER*2       20 *
LSA     INTEGER*2       24 *
LSJ     INTEGER*2       28 *
N       INTEGER*2      164
NCDOF   INTEGER*2        6   /DATA   /
NCMOD   INTEGER*2        2   /DATA   /
NFGEN   INTEGER*2      156   /DATA   /
NGYRO   INTEGER*2        0   /DATA   /
NI      INTEGER*2      182   /DATA   /
NMODES  INTEGER*2      166
NN      INTEGER*2      184
NRC     INTEGER*2       36 *
NRDOF   INTEGER*2        8   /DATA   /
NRMOD   INTEGER*2        4   /DATA   /
NSA     INTEGER*2      158   /DATA   /
NSL     INTEGER*2      166   /DATA   /
NXTC    INTEGER*2      172   /DATA   /
RPM1    REAL           174   /DATA   /
S       REAL            40 *
TITL    CHAR*1          10   /DATA   /
WK      REAL*8          56 *
```

```
*ANSR.FOR*****************************************************************
C
C      ANSR ORGANIZES OUTPUT INFORMATION AND PLOTS,SAVES
C      THAT INFORMATION PER USER REQUEST.
C      ANSR CALLS SUBROUTINE ZSORT.
C
C*************************************************************************
C
       SUBROUTINE ANSR(Z,LDYN,RPM,W,ZRPM)
C
       REAL*8       Z(2,LDYN,1),WZ
C
       DIMENSION    ZRPM(1),W(2,1),NSA(4)
C
       LOGICAL      PLOT
C
       EXTERNAL    JCOL,JROW
C
       CHARACTER TITL(72,2)*1,TAPE4*10
C
       COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
      +                NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
      +                IPLTF,IPRT2,IPRT3
C
       COMMON     /RANGE/XR(5),YR(5)
C
C-----COMPRESS THE EIGENVALUE AND EIGENVECTOR ARRAYS
```

```
      C      (CORRECTING FOR REAL ROOTS)
      C
             N  =  NRMOD + NCMOD
             K=1
      C
             DO 20  I = 1,N
1             IFIX=I*2-K
1             IF(W(2,IFIX).EQ.0.) THEN
1              IF(K.EQ.1) THEN
1               K=2
1              ELSE
1               K=1
1               IFIX=I*2-K
1              ENDIF
1             ENDIF
1             W(1,I)=W(1,IFIX)
1             W(2,I)=W(2,IFIX)
1             DO 20 J=1,N
2              Z(1,J,I)=Z(1,J+N,IFIX)
2         20   Z(2,J,I)=Z(2,J+N,IFIX)
      C
      C-----SORT THE ARRAYS WITH RESPECT TO FREQUENCY
      C
             N2=N
      C
             DO 9 I=1,N-1
1              N2=N2-1
1              DO 9 J=1,N2
2                IF(W(2,J).LE.W(2,J+1)) GOTO 9
2                WZ=W(1,J+1)
2                W(1,J+1)=W(1,J)
2                W(1,J)=WZ
2                WZ=W(2,J+1)
2                W(2,J+1)=W(2,J)
2                W(2,J)=WZ
2                DO 8 K=1,N
3                  WZ=Z(1,K,J+1)
3                  Z(1,K,J+1)=Z(1,K,J)
3                  Z(1,K,J)=WZ
3                  WZ=Z(2,K,J+1)
3                  Z(2,K,J+1)=Z(2,K,J)
3          8       Z(2,K,J)=WZ
2          9 CONTINUE
      C
      C-----DETERMINE WHETHER THE MODE SHAPES SHOULD BE STORED FOR
      C      PRINTING/PLOTTING LATER
      C
             PLOT=.FALSE.
             IF(IPRT3.EQ.0.OR.IPRT.EQ.0) THEN
               J=1
          24   IF(J.LE.KRPM) THEN
                 IF(RPM.EQ.ZRPM(J)) PLOT=.TRUE.
                 J=J+1
                 GOTO 24
               ENDIF
             ENDIF
      C
      C-----STORE AND PLOT RESULTS
      C
```

```
                WRITE(TAPE4,8000) RPM
                I=69
                J=21
                CALL QPTXT(10,TAPE4,3,I,J)
                WRITE(6,7)
          C.....FOR EACH MODE, WRITE RESULTS AND PLOT ON THE CRT
                DO 35 I=1,N
1                 C = W(2,I) * 9.54929
1                 WRITE(6,30) I,(W(J,I),J=1,2),C
1         C.......IF IN CRT WINDOW, PLOT EIGENVALUE
1                 IF(C.LE.YR(5).AND.W(1,I).GE.XR(3).AND.W(1,I).LE.XR(5)) THEN
1                   CALL QRAST(JCOL(W(1,I))-1,JROW(C),3,2)
1         C.........IF FIRST RPM STEP, PLOT SPECIAL SYMBOL
1                   IF(RPM.EQ.RPM1.OR.RPM.EQ.RPM1+DRPM*NI) THEN
1                     CALL QSPNT(JCOL(W(1,I)),JROW(C)+1,3)
1                     CALL QSPNT(JCOL(W(1,I)),JROW(C)-1,3)
1                   ENDIF
1         C.........IF LAST RPM STEP, PLOT SPECIAL SYMBOL
1                   IF(RPM.EQ.RPM1+DRPM*NI) THEN
1                     CALL QSPNT(JCOL(W(1,I))-1,JROW(C)+1,3)
1                     CALL QSPNT(JCOL(W(1,I))-1,JROW(C)-1,3)
1                     CALL QSPNT(JCOL(W(1,I))+1,JROW(C)+1,3)
1                     CALL QSPNT(JCOL(W(1,I))+1,JROW(C)-1,3)
1                     CALL QSPNT(JCOL(W(1,I)),JROW(C),0)
1                   ENDIF
1                 ENDIF
1                 IF(.NOT.PLOT) GOTO 35
1         C.......IF THIS MODE SHAPE IS TO BE PLOTTED, STORE SHAPE
1                 IF(C.GE.FLOW.AND.C.LE.FHIGH) WRITE(3) RPM,I,(W(J,I),J=1,2),
1              +                                ((SNGL(Z(J,L,I)),J=1,2),L=1,N)
1            35 CONTINUE
          C.....STORE EIGENVALUES
                WRITE(2) ((W(J,I),J=1,2),I=1,N)
          C
          C-----FORMAT STATEMENTS
          C
              7 FORMAT(//5X,'MODE',11X,'REAL',14X,'IMAGINARY',14X,'CPM'/)
             30 FORMAT(4X,I4,3(5X,1PE15.5))
           8000 FORMAT(4H TO ,F6.0)
          C
                END
```

Name	Type	Offset	P	Class
A	REAL	0	*	
C	REAL	24	*	
D	REAL*8	40	*	
G	REAL	16	*	
I	INTEGER*2	424		
J	INTEGER*2	432		
K	INTEGER*2	440		
L	INTEGER*2	448		
LRA	INTEGER*2	4	*	
LRC	INTEGER*2	28	*	
LRD	INTEGER*2	44	*	
LRG	INTEGER*2	20	*	
M	INTEGER*2	456		
NCA	INTEGER*2	12	*	
NCC	INTEGER*2	36	*	

```
D Line# 1      7
NRA     INTEGER*2          8 *
NRC     INTEGER*2         32 *
WK      REAL*8            48 *


    400 $LIST

Name    Type             Size   Class

ANSR                            SUBROUTINE
DATA                      198   COMMON
EIGRF                           SUBROUTINE
FORT03                          SUBROUTINE
JCOL    INTEGER*2               FUNCTION
JROW    INTEGER*2               FUNCTION
MATX                            SUBROUTINE
QPTXT                           SUBROUTINE
QRAST                           SUBROUTINE
QSPNT                           SUBROUTINE
RANGE                      40   COMMON
TRANS                           SUBROUTINE


 Pass One     No Errors Detected
             400 Source Lines
```

```
$STORAGE:2
C   ROUTINE NAME         - EIGRF.FOR (FROM IMSL ROUTINE EIGRF)
C
C-----------------------------------------------------------------------
C
C   COMPUTER             - IBM/SINGLE
C
C   LATEST REVISION      - MARCH 19, 1984
C
C   PURPOSE              - EIGENVALUES AND (OPTIONALLY) EIGENVECTORS OF
C                            A REAL GENERAL MATRIX IN FULL STORAGE MODE
C
C   USAGE                - CALL EIGRF (A,N,IA,W,Z,IZ,WK,IER)
C
C   ARGUMENTS    A       - THE INPUT REAL GENERAL MATRIX OF ORDER N
C                            WHOSE EIGENVALUES AND EIGENVECTORS ARE
C                            TO BE COMPUTED. INPUT A IS DESTROYED.
C                N       - THE INPUT ORDER OF THE MATRIX A.
C                IA      - THE INPUT ROW DIMENSION OF MATRIX A EXACTLY
C                            AS SPECIFIED IN THE DIMENSION STATEMENT IN
C                            THE CALLING PROGRAM.
C                W       - THE OUTPUT COMPLEX VECTOR OF LENGTH N,
C                            CONTAINING THE EIGENVALUES OF A.
C                          NOTE - THE ROUTINE TREATS W AS A REAL VECTOR
C                            OF LENGTH 2*N. AN APPROPRIATE EQUIVALENCE
C                            STATEMENT MAY BE REQUIRED. SEE DOCUMENT
C                            EXAMPLE.
C                Z       - THE OUTPUT N BY N COMPLEX MATRIX CONTAINING
C                            THE EIGENVECTORS OF A.
C                            THE EIGENVECTOR IN COLUMN J OF Z CORRES-
C                            PONDS TO THE EIGENVALUE W(J).
C                          NOTE - THE ROUTINE TREATS Z AS A REAL VECTOR
C                            OF LENGTH 2*N*N. AN APPROPRIATE EQUIVALENCE
C                            STATEMENT MAY BE REQUIRED. SEE DOCUMENT
C                            EXAMPLE.
C                IZ      - THE INPUT ROW DIMENSION OF MATRIX Z EXACTLY
C                            AS SPECIFIED IN THE DIMENSION STATEMENT IN
C                            THE CALLING PROGRAM. IZ MUST BE GREATER
C                            THAN OR EQUAL TO N.
C                WK      - WORK AREA, THE LENGTH OF WK IS AT LEAST 2*N.
C                IER     - ERROR PARAMETER. (OUTPUT)
C                          TERMINAL ERROR
C                            IER = 128+J, INDICATES THAT EQRH3F FAILED
C                            TO CONVERGE ON EIGENVALUE J. EIGENVALUES
C                            J+1,J+2,...,N HAVE BEEN COMPUTED CORRECTLY.
C                            EIGENVALUES 1,...,J ARE SET TO ZERO.
C                            EIGENVECTORS ARE SET TO ZERO.
C
C   REQD. SUBROUTINES    - EBALAF,EBBCKF,EHBCKF,EHESSF,EQRH3F
C
C-----------------------------------------------------------------------
C
      SUBROUTINE EIGRF   (A,N,IA,W,Z,IZ,WK,IER)
C                                  SPECIFICATIONS FOR ARGUMENTS
      INTEGER            N,IA,IZ,IER
      REAL*8             A(IA,1),WK(N,1),Z(1)
      REAL*4             W(1)
C                                  SPECIFICATIONS FOR LOCAL VARIABLES
      INTEGER             IZ2,K,L,I,N1,N2,II,JJ,NP1,NPI,JW,J,
```

```
D Line# 1      7                                  Microsoft FORTRAN77 V3.13 8/05/83
               *                   IS,IG,IGZ
                REAL*8             ZERO,ONE
                DATA               ZERO,ONE/0.0,1.0/
          C                                  INITIALIZE ERROR PARAMETERS
          C                                  FIRST EXECUTABLE STATEMENT
                IER = 0
                IZ2 = IZ+IZ
          C                                  PACK A INTO AN N BY N ARRAY
                K = 1
                L = 1
                DO 20 J=1,N
1                  DO 20 I=1,N
2                     A(K,L) = A(I,J)
2                     K = K+1
2                     IF (K .GT. IA) K = 1
2                     IF (K .EQ. 1) L = L+1
2            20 CONTINUE
                N1 = 1
                N2 = N1+1
          C                                  BALANCE THE INPUT A
                CALL EBALAF (A,N,WK,K,L)
          C                                  IF L = 0, A IS ALREADY IN HESSENBERG
          C                                    FORM
                CALL EHESSF (A,K,L,N,WK(1,N2))
          C                                  SET Z IDENTITY MATRIX
                II = 1
                JJ = 1
                NP1 = N+1
                DO 30 I=1,N
1                  DO 25 J=1,N
2                     Z(II) = ZERO
2                     II = II+1
2            25    CONTINUE
1                  Z(JJ) = ONE
1                  JJ = JJ+NP1
1            30 CONTINUE
                CALL EHBCKF (Z,A,WK(1,N2),N,K,L)
                CALL EQRH3F (A,N,K,L,W(1),W(N+1),Z,IER)
                IF (IER .GT. 128) GO TO 40
                CALL EBBCKF (WK,Z,K,L,N)
          C                                  CONVERT W (EIGENVALUES) TO COMPLEX
          C                                    FORMAT
             40 DO 45 I=1,N
1                  NPI = N+I
1                  WK(I,N1) = W(NPI)
1            45 CONTINUE
                JW = N+N
                J = N
                DO 50 I=1,N
1                  W(JW-1) = W(J)
1                  W(JW) = WK(J,N1)
1                  JW = JW-2
1                  J = J-1
1            50 CONTINUE
          C                                  CONVERT Z (EIGENVECTORS) TO COMPLEX
          C                                    FORMAT Z(IZ,N)
                J = N
             60 IF (J .LT. 1) GO TO 9000
                IF (W(J+J) .EQ. ZERO) GO TO 75
```

```
D Line# 1       7                                  Microsoft FORTRAN77 V3.13 8/05/83
      C                                         MOVE PAIR OF COMPLEX CONJUGATE
      C                                           EIGENVECTORS
               IS = IZ2*(J-1)+1
               IG = N*(J-2)+1
               IGZ = IG+N
      C                                         MOVE COMPLEX CONJUGATE EIGENVECTOR
               DO 65 I=1,N
1                 Z(IS) = Z(IG)
1                 Z(IS+1) = -Z(IGZ)
1                 IS = IS+2
1                 IG = IG+1
1                 IGZ = IGZ+1
1           65 CONTINUE
      C                                         MOVE COMPLEX EIGENVECTOR
               IS = IZ2*(J-2)+1
               IG = IS+IZ2
               DO 70 I=1,N
1                 Z(IS) = Z(IG)
1                 Z(IS+1) = -Z(IG+1)
1                 IS = IS+2
1                 IG = IG+2
1           70 CONTINUE
               J = J-2
               GO TO 60
      C                                         MOVE REAL EIGENVECTOR
            75 IS = IZ2*(J-1)+N+N
               IG = N*J
               DO 80 I=1,N
1                 Z(IS-1) = Z(IG)
1                 Z(IS) = ZERO
1                 IS = IS-2
1                 IG = IG-1
1           80 CONTINUE
               J = J-1
               GO TO 60
      C                                         Z IS NOW IN COMPLEX FORMAT Z(IZ,N).
          9000 CONTINUE
               END
```

Name	Type	Offset	P	Class
A	REAL*8	0	*	
I	INTEGER*2	32		
IA	INTEGER*2	8	*	
IER	INTEGER*2	28	*	
IG	INTEGER*2	80		
IGZ	INTEGER*2	82		
II	INTEGER*2	44		
IS	INTEGER*2	78		
IZ	INTEGER*2	20	*	
IZ2	INTEGER*2	18		
J	INTEGER*2	24		
JJ	INTEGER*2	46		
JW	INTEGER*2	70		
K	INTEGER*2	20		
L	INTEGER*2	22		
N	INTEGER*2	4	*	
N1	INTEGER*2	40		
N2	INTEGER*2	42		

```
D Line# 1      7                              Microsoft FORTRAN77 V3.13 8/05/83
NP1     INTEGER*2          48
NPI     INTEGER*2          68
ONE     REAL*8             10
W       REAL               12 *
WK      REAL*8             24 *
Z       REAL*8             16 *
ZERO    REAL*8              2


       C
       C-----------------------------------------------------------------------
       C
                SUBROUTINE EBALAF (A,N,D,K,L)
       C                                         SPECIFICATIONS FOR ARGUMENTS
                INTEGER            N,K,L
                REAL*8             A(N,1),D(1)
       C                                         SPECIFICATIONS FOR LOCAL VARIABLES
                INTEGER            L1,K1,K1P1,K11,JJ,J,I,LL,NOCONV
                REAL*8             R,C,F,G,B,S,B2,ONE,ZERO,P95
                DATA               B/16.0/,B2/256.0/
                DATA               ZERO/0.0/,ONE/1.0/,P95/.95/
       C                                         REDUCE NORM A BY DIAGONAL SIMILARITY
       C                                         TRANSFORMATION STORED IN D
       C                                         FIRST EXECUTABLE STATEMENT
                L1 = 1
                K1 = N
       C                                         SEARCH FOR ROWS ISOLATING AN EIGEN-
       C                                         VALUE AND PUSH THEM DOWN
              5 K1P1 = K1+1
                IF (K1.LT.1) GO TO 35
                K11=K1
                DO 30 JJ=1,K11
1                  J = K1P1-JJ
1                  R = ZERO
1                  DO 10 I=1,K1
2                     IF (I.EQ.J) GO TO 10
2                     R=R+DABS(A(J,I))
2            10    CONTINUE
1                  IF (R.NE.ZERO) GO TO 30
1                  D(K1) = J
1                  IF (J.EQ.K1) GO TO 25
1                  DO 15 I=1,K1
2                     F = A(I,J)
2                     A(I,J) = A(I,K1)
2                     A(I,K1) = F
2            15    CONTINUE
1                  DO 20 I=L1,N
2                     F = A(J,I)
2                     A(J,I) = A(K1,I)
2                     A(K1,I) = F
2            20    CONTINUE
1            25    K1 = K1-1
1                  GO TO 5
1            30 CONTINUE
       C                                         SEARCH FOR COLUMNS ISOLATING AN
       C                                         EIGENVALUE AND PUSH THEM LEFT
             35 IF (K1.LT.L1) GO TO 65
                LL = L1
                DO 60 J=LL,K1
```

```
D Line# 1       7
1                C = ZERO
1                DO 40 I=L1,K1
2                   IF (I.EQ.J) GO TO 40
2                   C = C+DABS(A(I,J))
2          40    CONTINUE
1                IF (C.NE.ZERO) GO TO 60
1                D(L1) = J
1                IF (J.EQ.L1) GO TO 55
1                DO 45 I=1,K1
2                   F = A(I,J)
2                   A(I,J) = A(I,L1)
2                   A(I,L1) = F
2          45    CONTINUE
1                DO 50  I=L1,N
2                   F = A(J,I)
2                   A(J,I) = A(L1,I)
2                   A(L1,I) = F
2          50    CONTINUE
1          55    L1 = L1+1
1                GO TO 35
1          60 CONTINUE
        C                                        NOW BALANCE THE SUBMATRIX IN ROWS
        C                                        L1 THROUGH K1
           65 K = L1
              L = K1
              IF (K1.LT.L1) GO TO 75
              DO 70  I=L1,K1
1                D(I) = ONE
1          70 CONTINUE
           75 NOCONV = 0
              IF (K1.LT.L1) GO TO 120
              DO 115 I=L1,K1
1                C = ZERO
1                R = ZERO
1                DO 80 J=L1,K1
2                   IF (J.EQ.I) GO TO 80
2                   C = C+DABS(A(J,I))
2                   R = R+DABS(A(I,J))
2          80    CONTINUE
1                G = R/B
1                F = ONE
1                S = C+R
1          85    IF (C.GE.G) GO TO 90
1                F = F * B
1                C = C*B2
1                GO TO 85
1          90    G = R*B
1          95    IF (C.LT.G) GO TO 100
1                F = F/B
1                C = C/B2
1                GO TO 95
1       C                                        NOW BALANCE
1         100    IF ((C+R)/F.GE.P95*S) GO TO 115
1                G = ONE/F
1                D(I) = D(I)*F
1                NOCONV = 1
1                DO 105 J=L1,N
2                   A(I,J) = A(I,J)*G
2         105    CONTINUE
```

```
D Line# 1      7                                    Microsoft FORTRAN77 V3.13 8/05/83
1                   DO 110 J=1,K1
2                      A(J,I) = A(J,I)*F
2         110       CONTINUE
1         115 CONTINUE
          120 IF (NOCONV.EQ.1) GO TO 75
              END
```

Name	Type	Offset	P	Class
A	REAL*8	0	*	
B	REAL*8	102		
B2	REAL*8	110		
C	REAL*8	204		
D	REAL*8	8	*	
DABS				INTRINSIC
F	REAL*8	182		
G	REAL*8	250		
I	INTEGER*2	168		
J	INTEGER*2	158		
JJ	INTEGER*2	150		
K	INTEGER*2	12	*	
K1	INTEGER*2	144		
K11	INTEGER*2	148		
K1P1	INTEGER*2	146		
L	INTEGER*2	16	*	
L1	INTEGER*2	142		
LL	INTEGER*2	196		
N	INTEGER*2	4	*	
NOCONV	INTEGER*2	236		
ONE	REAL*8	126		
P95	REAL*8	134		
R	REAL*8	160		
S	REAL*8	258		
ZERO	REAL*8	118		

```
     C
     C-----------------------------------------------------------------------
     C
              SUBROUTINE EBBCKF (D,Z,K,L,N)
     C                                             SPECIFICATIONS FOR ARGUMENTS
              INTEGER             K,L,N
              REAL*8              D(1)
              REAL*8              Z(N,1)
     C                                             SPECIFICATIONS FOR LOCAL VARIABLES
              INTEGER             I,J,KM1,II,JJ,LP1
              REAL*8              S
     C                                             COLUMN SCALE Z BY APPROPRIATE D VALUE
     C                                             FIRST EXECUTABLE STATEMENT
              IF(L.EQ.0) GO TO 6
              DO 5 I=K,L
1                S=D(I)
1                DO 5 J=1,N
2                   Z(I,J)=Z(I,J)*S
2           5 CONTINUE
     C                                             INTERCHANGE ROWS IF PERMUTATIONS
     C                                               OCCURRED IN EBALAF
            6 IF (K .EQ. 1) GO TO 20
              KM1=K-1
```

```
D Line# 1       7                                   Microsoft FORTRAN77 V3.13 8/05/83
                DO 15 I=1,KM1
1                  II=K-I
1                  JJ=D(II)
1                  IF (II .EQ. JJ) GO TO 15
1                  DO 10 J=1,N
2                     S=Z(II,J)
2                     Z(II,J)=Z(JJ,J)
2                     Z(JJ,J)=S
2          10      CONTINUE
1          15   CONTINUE
           20   IF (L .EQ. N) GO TO 35
                LP1=L+1
                DO 30 II=LP1,N
1                  JJ=D(II)
1                  IF (II .EQ. JJ) GO TO 30
1                  DO 25 J=1,N
2                     S=Z(II,J)
2                     Z(II,J)=Z(JJ,J)
2                     Z(JJ,J)=S
2          25      CONTINUE
1          30   CONTINUE
           35   RETURN
                END
```

Name	Type	Offset	P	Class
D	REAL*8	0	*	
I	INTEGER*2	278		
II	INTEGER*2	310		
J	INTEGER*2	294		
JJ	INTEGER*2	312		
K	INTEGER*2	8	*	
KM1	INTEGER*2	302		
L	INTEGER*2	12	*	
LP1	INTEGER*2	320		
N	INTEGER*2	16	*	
S	REAL*8	286		
Z	REAL*8	4	*	

```
      C
      C-----------------------------------------------------------------------
      C
            SUBROUTINE EHBCKF (Z,H,D,N,K,L)
      C                                   SPECIFICATIONS FOR ARGUMENTS
            INTEGER            N,K,L
            REAL*8             H(N,1),D(1)
            REAL*8             Z(N,1)
      C                                   SPECIFICATIONS FOR LOCAL VARIABLES
            INTEGER            LM2,KI,LTEMP,M,MA,MP2,I,J
            REAL*8             T,TINV,ZERO,ONE,G
            DATA               ZERO,ONE/0.0,1.0/
      C                                   FIRST EXECUTABLE STATEMENT
            LM2=L-2
            IF(LM2.LT.K) GO TO 9005
            LTEMP=LM2+K
            DO 30 KI=K,LM2
1              M=LTEMP-KI
1              MA=M+1
```

```
D Line# 1       7                                      Microsoft FORTRAN77 V3.13 8/05/83
1    337            T=H(MA,M)
1    338            IF(T.EQ.ZERO) GO TO 30
1    339            T=T*D(MA)
1    340            MP2=M+2
1    341            IF(MP2.GT.L) GO TO 10
1    342            DO 5 I=MP2,L
2    343               D(I)=H(I,M)
2    344      5     CONTINUE
1    345     10     IF(MA.GT.L) GO TO 30
1    346            TINV = ONE / T
1    347            DO 25 J=1,N
2    348               G=ZERO
2    349               DO 15 I=MA,L
3    350                  G=G+D(I)*Z(I,J)
3    351     15        CONTINUE
2    352               G = G*TINV
2    353               DO 20 I=MA,L
3    354                  Z(I,J)=Z(I,J)+G*D(I)
3    355     20        CONTINUE
2    356     25     CONTINUE
1    357     30  CONTINUE
     358   9005  RETURN
     359         END

Name    Type        Offset P Class

D       REAL*8           8 *
G       REAL*8         400
H       REAL*8           4 *
I       INTEGER*2      376
J       INTEGER*2      392
K       INTEGER*2       16 *
KI      INTEGER*2      354
L       INTEGER*2       20 *
LM2     INTEGER*2      350
LTEMP   INTEGER*2      352
M       INTEGER*2      362
MA      INTEGER*2      364
MP2     INTEGER*2      374
N       INTEGER*2       12 *
ONE     REAL*8         342
T       REAL*8         366
TINV    REAL*8         384
Z       REAL*8           0 *
ZERO    REAL*8         334


     360 C-----------------------------------------------------------------------
     361 C
     362         SUBROUTINE EHESSF (A,K,L,N,D)
     363 C                                     SPECIFICATIONS FOR ARGUMENTS
     364         INTEGER             K,L,N
     365         REAL*8              A(N,N),D(N)
     366 C                                     SPECIFICATIONS FOR LOCAL VARIABLES
     367         INTEGER             LA,KP1,M,I,MP,II,J,JJ
     368         REAL*8              F,G,H,SCALE,ZERO
     369         DATA                ZERO/0.0/
     370 C                                     FIRST EXECUTABLE STATEMENT
     371         LA = L - 1
```

```
D Line# 1      7
               KP1 = K + 1
               IF (LA .LT. KP1) GO TO 50
               DO 45 M = KP1, LA
1                 H = ZERO
1                 D(M) = ZERO
1                 SCALE = ZERO
1       C                                         SCALE COLUMN
1                 DO 5 I = M, L
2                    SCALE = SCALE + DABS(A(I,M-1))
2           5     CONTINUE
1                 IF (SCALE .EQ. ZERO ) GO TO 45
1                 MP = M + L
1       C                                         DO 10 I=L,M,-1
1                 DO 10 II = M, L
2                    I = MP - II
2                    D(I) = A(I,M-1) / SCALE
2                    H = H + D(I) * D(I)
2          10     CONTINUE
1                 G = -DSIGN(DSQRT(H),D(M))
1                 H = H - D(M) * G
1                 D(M) = D(M) - G
1       C                                         FORM (I-(U*UT)/H) * A
1                 DO 25 J = M,N
2                    F = ZERO
2       C                                         DO 15 I=L,M,-1
2                    DO 15 II = M, L
3                       I = MP - II
3                       F = F + D(I) * A(I,J)
3          15        CONTINUE
2                    F = F / H
2                    DO 20 I = M, L
3                       A(I,J) = A(I,J) - F * D(I)
3          20        CONTINUE
2          25     CONTINUE
1       C                                         FORM (I-(U*UT)/H)*A*(I-(U*UT)/H)
1                 DO 40 I = 1,L
2                    F = ZERO
2       C                                         DO 30 J=L,M,-1
2                    DO 30 JJ = M, L
3                       J = MP - JJ
3                       F = F + D(J) * A(I,J)
3          30        CONTINUE
2                    F = F / H
2                    DO 35 J = M, L
3                       A(I,J) = A(I,J) - F * D(J)
3          35        CONTINUE
2          40     CONTINUE
1                 D(M) = SCALE * D(M)
1                 A(M,M-1) = SCALE * G
1          45  CONTINUE
           50  RETURN
               END
```

```
Name     Type          Offset P Class

A        REAL*8             0 *
D        REAL*8            16 *
DABS                             INTRINSIC
DSIGN                            INTRINSIC
```

```
DSQRT                          INTRINSIC
F        REAL*8          490
G        REAL*8          474
H        REAL*8          440
I        INTEGER*2       456
II       INTEGER*2       466
J        INTEGER*2       482
JJ       INTEGER*2       516
K        INTEGER*2         4 *
KP1      INTEGER*2       430
L        INTEGER*2         8 *
LA       INTEGER*2       428
M        INTEGER*2       432
MP       INTEGER*2       464
N        INTEGER*2        12 *
SCALE    REAL*8          448
ZERO     REAL*8          420
```

```
      C-----------------------------------------------------------------------
      C
            SUBROUTINE EQRH3F (H,N,K,L,WR,WI,Z,IER)
      C                                   SPECIFICATIONS FOR ARGUMENTS
            INTEGER            N,K,L,IER
            REAL*8             H(N,N),Z(N,N)
            REAL*4             WR(N),WI(N)
      C                                   SPECIFICATIONS FOR LOCAL VARIABLES
            INTEGER            I,IEN,ITS,IENM2,NPL,LL,LB,NAML,MM,M,MP2,KA,NA,
           *                   J,JJ
            REAL*8             RDELP,P4,P5,P7,ZERO,ONE,T,W,S,F,X,Y,ZZ,Q,R,
           *                   RNORM,RA,SA,VR,VI
            LOGICAL            NOTLAS
            DATA               RDELP/7.105427357601001859E-15/
            DATA               P4/0.4375/,P5/0.5/,P7/0.75/,ZERO/0.0/,ONE/1.0/
      C                                   FIRST EXECUTABLE STATEMENT
            IER = 0
      C                                   STORE ROOTS ISOLATED BY EBALAF
            DO 5 I=1,N
1              IF (I.GE.K.AND.I.LE.L) GO TO 5
1              WR(I) = H(I,I)
1              WI(I) = ZERO
1         5 CONTINUE
            IEN = L
            T = ZERO
      C                                   SEARCH FOR NEXT EIGENVALUES
         10 IF (IEN.LT.K) GO TO 140
            ITS = 0
            NA = IEN-1
            IENM2 = NA-1
      C                                   LOOK FOR SINGLE SMALL SUB-DIAGONA
      C                                   ELEMENT
         15 NPL = IEN+K
            DO 20 LL=K,IEN
1              LB = NPL-LL
1              IF (LB.EQ.K) GO TO 25
1              IF (DABS(H(LB,LB-1)).LE.RDELP*(DABS(H(LB-1,LB-1))
1          1      +DABS(H(LB,LB)))) GO TO 25
1        20 CONTINUE
      C
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.13 8/05/83
            25 X = H(IEN,IEN)
               IF (LB.EQ.IEN) GO TO 105
               Y = H(NA,NA)
               W = H(IEN,NA)*H(NA,IEN)
               IF (LB.EQ.NA) GO TO 110
               IF (ITS.EQ.30) GO TO 255
         C                                        FORM SHIFT
               IF (ITS.NE.10.AND.ITS.NE.20) GO TO 35
               T = T+X
               DO 30 I=K,IEN
 1                H(I,I) = H(I,I)-X
 1          30 CONTINUE
               S = DABS(H(IEN,NA))+DABS(H(NA,IENM2))
               X = P7*S
               Y = X
               W = -P4*S*S
            35 ITS = ITS+1
         C                                      LOOK FOR TWO CONSECUTIVE SMALL
         C                                       SUB-DIAGONAL ELEMENTS
               NAML = IENM2+LB
               DO 40 MM=LB,IENM2
 1                M = NAML-MM
 1                ZZ = H(M,M)
 1                R = X-ZZ
 1                S = Y-ZZ
 1                P = (R*S-W)/H(M+1,M)+H(M,M+1)
 1                Q = H(M+1,M+1)-ZZ-R-S
 1                R = H(M+2,M+1)
 1                S = DABS(P)+DABS(Q)+DABS(R)
 1                P = P/S
 1                Q = Q/S
 1                R = R/S
 1                IF (M.EQ.LB) GO TO 45
 1                IF (DABS(H(M,M-1))*(DABS(Q)+DABS(R)).LE.RDELP*DABS(P)*
 1        1          (DABS(H(M-1,M-1))+DABS(ZZ)+DABS(H(M+1,M+1)))) GO TO 45
 1          40 CONTINUE
            45 MP2 = M+2
               DO 50 I=MP2,IEN
 1                H(I,I-2) = ZERO
 1                IF (I.EQ.MP2) GO TO 50
 1                H(I,I-3) = ZERO
 1          50 CONTINUE
         C                                        DOUBLE QR STEP INVOLVING ROWS
         C                                        L TO EN AND COLUMNS M TO EN
               DO 100 KA=M,NA
 1                NOTLAS = KA.NE.NA
 1                IF (KA.EQ.M) GO TO 55
 1                P = H(KA,KA-1)
 1                Q = H(KA+1,KA-1)
 1                R = ZERO
 1                IF (NOTLAS) R = H(KA+2,KA-1)
 1                X = DABS(P)+DABS(Q)+DABS(R)
 1                IF (X.EQ.ZERO) GO TO 100
 1                P = P/X
 1                Q = Q/X
 1                R = R/X
 1          55    CONTINUE
 1                S = DSIGN(DSQRT(P*P+Q*Q+R*R),P)
 1                IF (KA.EQ.M) GO TO 60
```

```
1            H(KA,KA-1) = -S*X
1            GO TO 65
1       60   IF (LB.NE.M) H(KA,KA-1) = -H(KA,KA-1)
1       65   P = P+S
1            X = P/S
1            Y = Q/S
1            ZZ = R/S
1            Q = Q/P
1            R = R/P
1     C                                    ROW MODIFICATION
1            DO 75 J=KA,N
2               P = H(KA,J)+Q*H(KA+1,J)
2               IF (.NOT.NOTLAS) GO TO 70
2               P = P+R*H(KA+2,J)
2               H(KA+2,J) = H(KA+2,J)-P*ZZ
2       70      H(KA+1,J) = H(KA+1,J)-P*Y
2               H(KA,J) = H(KA,J)-P*X
2       75   CONTINUE
1            J = MINO(IEN,KA+3)
1     C                                    COLUMN MODIFICATION
1            DO 85 I=1,J
2               P = X*H(I,KA)+Y*H(I,KA+1)
2               IF (.NOT.NOTLAS) GO TO 80
2               P = P+ZZ*H(I,KA+2)
2               H(I,KA+2) = H(I,KA+2)-P*R
2       80      H(I,KA+1) = H(I,KA+1)-P*Q
2               H(I,KA) = H(I,KA)-P
2       85   CONTINUE
1     C                                    ACCUMULATE TRANSFORMATIONS
1            DO 95 I=K,L
2               P = X*Z(I,KA)+Y*Z(I,KA+1)
2               IF (.NOT.NOTLAS) GO TO 90
2               P = P+ZZ*Z(I,KA+2)
2               Z(I,KA+2) = Z(I,KA+2)-P*R
2       90      Z(I,KA+1) = Z(I,KA+1)-P*Q
2               Z(I,KA) = Z(I,KA)-P
2       95   CONTINUE
1      100 CONTINUE
           GO TO 15
      C                                    ONE ROOT FOUND
       105 H(IEN,IEN) = X+T
           WR(IEN) = H(IEN,IEN)
           WI(IEN) = ZERO
           IEN = NA
           GO TO 10
      C                                    TWO ROOTS FOUND
       110 P = (Y-X)*P5
           Q = P*P+W
           ZZ = DSQRT(DABS(Q))
           H(IEN,IEN) = X+T
           X = H(IEN,IEN)
           H(NA,NA) = Y+T
           IF (Q.LT.ZERO) GO TO 130
      C                                    REAL PAIR
           ZZ = P+DSIGN(ZZ,P)
           WR(NA) = X+ZZ
           WR(IEN) = WR(NA)
           IF (ZZ.NE.ZERO) WR(IEN) = X-W/ZZ
           WI(NA) = ZERO
```

```
      WI(IEN) = ZERO
      X = H(IEN,NA)
      R = DSQRT(X*X+ZZ*ZZ)
      P = X/R
      Q = ZZ/R
C                                 ROW MODIFICATION
      DO 115 J=NA,N
         ZZ = H(NA,J)
         H(NA,J) = Q*ZZ+P*H(IEN,J)
         H(IEN,J) = Q*H(IEN,J)-P*ZZ
  115 CONTINUE
C                                 COLUMN MODIFICATION
      DO 120 I=1,IEN
         ZZ = H(I,NA)
         H(I,NA) = Q*ZZ+P*H(I,IEN)
         H(I,IEN) = Q*H(I,IEN)-P*ZZ
  120 CONTINUE
C                                 ACCUMULATE TRANSFORMATIONS
      DO 125 I=K,L
         ZZ = Z(I,NA)
         Z(I,NA) = Q*ZZ+P*Z(I,IEN)
         Z(I,IEN) = Q*Z(I,IEN)-P*ZZ
  125 CONTINUE
      GO TO 135
C                                 COMPLEX PAIR
  130 WR(NA) = X+P
      WR(IEN) = X+P
      WI(NA) = ZZ
      WI(IEN) = -ZZ
  135 IEN = IENM2
      GO TO 10
C                                 ALL ROOTS FOUND, NOW
C                                 BACKSUBSTITUTE
  140 RNORM = ZERO
      KA = 1
      DO 150 I=1,N
         DO 145 J=KA,N
            RNORM = RNORM+DABS(H(I,J))
  145    CONTINUE
         KA = I
  150 CONTINUE
      IF (RNORM.EQ.ZERO) GO TO 9000
      DO 225 NN=1,N
         IEN = N+1-NN
         P = WR(IEN)
         Q = WI(IEN)
         NA = IEN-1
         IF (Q.GT.ZERO) GO TO 225
         IF (Q.LT.ZERO) GO TO 185
C                                 REAL VECTOR
         M = IEN
         H(IEN,IEN) = ONE
         IF (NA.EQ.0) GO TO 225
         DO 180 II=1,NA
            I = IEN-II
            W = H(I,I)-P
            R = H(I,IEN)
            IF (M.GT.NA) GO TO 160
            DO 155 J=M,NA
```

```
3               R = R+H(I,J)*H(J,IEN)
3     155     CONTINUE
2     160     IF (WI(I).GE.ZERO) GO TO 165
2             ZZ = W
2             S = R
2             GO TO 180
2     165     M = I
2             IF (WI(I).NE.ZERO) GO TO 170
2             T = W
2             IF (W.EQ.ZERO) T = RDELP*RNORM
2             H(I,IEN) = -R/T
2             GO TO 180
2   C                                    SOLVE REAL EQUATIONS
2     170     X = H(I,I+1)
2             Y = H(I+1,I)
2             Q = (WR(I)-P)*(WR(I)-P)+WI(I)*WI(I)
2             T = (X*S-ZZ*R)/Q
2             H(I,IEN) = T
2             IF (DABS(X).LE.DABS(ZZ)) GO TO 175
2             H(I+1,IEN) = (-R-W*T)/X
2             GO TO 180
2     175     H(I+1,IEN) = (-S-Y*T)/ZZ
2     180   CONTINUE
1   C                                    END REAL VECTOR
1           GO TO 225
1   C                                    LAST VECTOR COMPONENT CHOSEN
1   C                                      IMAGINARY SO THAT EIGENVECTOR
1   C                                      MATRIX IS TRIANGULAR
1     185   M = NA
1   C                                    COMPLEX VECTOR
1           IF (DABS(H(IEN,NA)).LE.DABS(H(NA,IEN))) GO TO 190
1           H(NA,NA) = Q/H(IEN,NA)
1           H(NA,IEN) = -(H(IEN,IEN)-P)/H(IEN,NA)
1           GO TO 195
1     190   CONTINUE
1         CALL CMPDIV(H(NA,NA),H(NA,IEN),ZERO,-H(NA,IEN),H(NA,NA)-P,Q)
1     195   H(IEN,NA) = ZERO
1           H(IEN,IEN) = ONE
1           IENM2 = NA-1
1           IF (IENM2.EQ.0) GO TO 225
1           DO 220 II=1,IENM2
2             I = NA-II
2             W = H(I,I)-P
2             RA = ZERO
2             SA = H(I,IEN)
2             DO 200 J=M,NA
3               RA = RA+H(I,J)*H(J,NA)
3               SA = SA+H(I,J)*H(J,IEN)
3     200     CONTINUE
2             IF (WI(I).GE.ZERO) GO TO 205
2             ZZ = W
2             R = RA
2             S = SA
2             GO TO 220
2     205     M = I
2             IF (WI(I).NE.ZERO) GO TO 210
2           CALL CMPDIV(H(I,NA),H(I,IEN),-RA,-SA,W,Q)
2             GO TO 220
2   C                                    SOLVE COMPLEX EQUATIONS
```

```
D Line# 1     7
2        210          X = H(I,I+1)
2                     Y = H(I+1,I)
2                     VR = (WR(I)-P)*(WR(I)-P)+WI(I)*WI(I)-Q*Q
2                     VI = (WR(I)-P)*Q
2                     VI = VI+VI
2                     IF (VR.EQ.ZERO.AND.VI.EQ.ZERO) VR = RDELP*RNORM
2             1          *(DABS(W)+DABS(Q)+DABS(X)+DABS(Y)+DABS(ZZ))
2                  CALL CMPDIV(H(I,NA),H(I,IEN),X*R-ZZ*RA+Q*SA,X*S-ZZ*SA-Q*RA,
2             1          VR,VI)
2                     IF (DABS(X).LE.DABS(ZZ)+DABS(Q)) GO TO 215
2                     H(I+1,NA) = (-RA-W*H(I,NA)+Q*H(I,IEN))/X
2                     H(I+1,IEN) = (-SA-W*H(I,IEN)-Q*H(I,NA))/X
2                     GO TO 220
2        215          CONTINUE
2                  CALL CMPDIV(H(I+1,NA),H(I+1,IEN),-R-Y*H(I,NA),-S-Y*H(I,IEN)
2             1          ,ZZ,Q)
2        220      CONTINUE
1            C                                   END COMPLEX VECTOR
1        225 CONTINUE
             C                                   END BACKSUBSTITUTION
             C                                   VECTORS OF ISOLATED ROOTS
                 DO 235 I=1,N
1                   IF (I.GE.K.AND.I.LE.L) GO TO 235
1                   DO 230 J=I,N
2                      Z(I,J) = H(I,J)
2        230        CONTINUE
1        235 CONTINUE
                 IF (L.EQ.0) GO TO 9000
             C                                   MULTIPLY BY TRANSFORMATION MATRIX
                 DO 250 JJ=K,N
1                   J = N+K-JJ
1                   M = MIN0(J,L)
1                   DO 245 I=K,L
2                      ZZ = ZERO
2                      DO 240 KA=K,M
3                         ZZ = ZZ+Z(I,KA)*H(KA,J)
3        240           CONTINUE
2                      Z(I,J) = ZZ
2        245        CONTINUE
1        250 CONTINUE
                 GO TO 9000
             C                                   NO CONVERGENCE AFTER 30 ITERATION
             C                                   SET ERROR INDICATOR  TO THE INDEX
             C                                   OF THE CURRENT EIGENVALUE
         255 IER = 128+IEN
             DO 260 I=1,IEN
1                   WR(I) = ZERO
1                   WI(I) = ZERO
1        260 CONTINUE
             DO 270 I=1,N
1                   DO 265 J=1,N
2                      Z(I,J) = ZERO
2        265        CONTINUE
1        270 CONTINUE
        9000 CONTINUE
             END
```

Name   Type   Offset P Class

```
D Line# 1       7
DABS                                  INTRINSIC
DSIGN                                 INTRINSIC
DSQRT                                 INTRINSIC
H        REAL*8             0 *
I        INTEGER*2        578
IEN      INTEGER*2        586
IENM2    INTEGER*2        600
IER      INTEGER*2         28 *
II       INTEGER*2        780
ITS      INTEGER*2        596
J        INTEGER*2        714
JJ       INTEGER*2        850
K        INTEGER*2          8 *
KA       INTEGER*2        704
L        INTEGER*2         12 *
LB       INTEGER*2        612
LL       INTEGER*2        604
M        INTEGER*2        662
MINO                                  INTRINSIC
MM       INTEGER*2        654
MP2      INTEGER*2        696
N        INTEGER*2          4 *
NA       INTEGER*2        598
NAML     INTEGER*2        652
NN       INTEGER*2        772
NOTLAS   LOGICAL*2        712
NPL      INTEGER*2        602
ONE      REAL*8           570
P        REAL*8           680
P4       REAL*8           538
P5       REAL*8           546
P7       REAL*8           554
Q        REAL*8           688
R        REAL*8           672
RA       REAL*8           800
RDELP    REAL*8           530
RNORM    REAL*8           752
S        REAL*8           644
SA       REAL*8           808
T        REAL*8           588
VI       REAL*8           830
VR       REAL*8           822
W        REAL*8           630
WI       REAL              20 *
WR       REAL              16 *
X        REAL*8           614
Y        REAL*8           622
Z        REAL*8            24 *
ZERO     REAL*8           562
ZZ       REAL*8           664
```

```
*CMPDIV*****************************************************************
C
C     S U B R O U T I N E   C M P D I V
C
C***********************************************************************
C
C     LATEST REVISION    - MARCH 19, 1984
```

```
C
C     PURPOSE             - DIVISION OF TWO COMPLEX NUMBERS
C
C     USAGE               - CALL CMPDIV(RR,RI,NR,NI,DR,DI)
C
C     ARGUMENTS   RR,RI   - REAL AND IMAGINARY PARTS OF RESULT
C                   NR,NI   - REAL AND IMAGINARY PARTS OF NUMERATOR
C                   DR,DI   - REAL AND IMAGINARY PARTS OF DENOMINATOR
C
C     REMARKS     RR AND RI WILL BE UNCHANGED IF THE MAGNITUDE OF THE
C                 DENOMINATOR IS ZERO (I.E., IF DR*DR + DI*DI = 0.0)
C
C*******************************************************************
C
      SUBROUTINE CMPDIV(RR,RI,NR,NI,DR,DI)
C
      REAL*8 RR,RI,NR,NI,DR,DI,D
C
      D  = DR*DR + DI*DI
      IF (D .EQ. 0.D0) RETURN
C
      RR = (NR*DR + NI*DI) / D
      RI = (NI*DR - NR*DI) / D
C
      END
```

Name	Type	Offset	P	Class
D	REAL*8	888		
DI	REAL*8	20	*	
DR	REAL*8	16	*	
NI	REAL*8	12	*	
NR	REAL*8	8	*	
RI	REAL*8	4	*	
RR	REAL*8	0	*	

Name	Type	Size	Class
CMPDIV			SUBROUTINE
EBALAF			SUBROUTINE
EBBCKF			SUBROUTINE
EHBCKF			SUBROUTINE
EHESSF			SUBROUTINE
EIGRF			SUBROUTINE
EQRH3F			SUBROUTINE

Pass One   No Errors Detected
787 Source Lines

```
*RSTABNP****************************************************************
C
C     P R O G R A M   R S T A B N P
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM (NO PLOTTING)
C
C***********************************************************************
C
C     SUBROUTINES UTILIZED IN RSTAB ARE CALLED
C     IN THE FOLLOWING ORDER:
C
C     FORT03 . . . . . . . . . . . . . . SOLVE FOR EIGENVALUES VS. RPM
C
C             MATX
C             EIGRF
C             ANSR
C
C**********************************************************************
C
$STORAGE:2
C
      PROGRAM RSTAB
C
      CHARACTER    TITL(72,2)*1,TAPE4*14
C
      INTEGER      NZ(4),NSA(4)
C
      REAL         RZ(6),RZ1(6)
C
      EQUIVALENCE (G,Z) , (TITL,TAPE4)
C
C* ARRAY DIMENSIONS MOST LIKELY TO CHANGE *****************************
C
C-----NOTES: DIMENSION WK TO (MAXO(LRDOF,LCDOF,2*LDYN))
C                      FUNC TO (NFGEN + 1)
C                      Z TO (2,LDYN,LDYN)
C                      W TO (2,LDYN)
C
      DIMENSION    AC(50,14),AR(115,16),WC(14),WR(16),ZETAC(14),X(10),
     +             SPEED(20,24),DMP(100,4),S(100,4),FG(20,24),
     +             G(115,115),FUNC(25),ZRPM(25),W(2,60)
C
      INTEGER      JFUN(160,2),NPT(24),NRC(100,2,4),IROT(20),ICASE(20)
C
      REAL*8       GBAR(16,16),A(60,60),WK(120),Z(2,60,60)
C
C**********************************************************************
C
      COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
     +                NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
     +                IPLTF,IPRT2,IPRT3
      COMMON    /MEM/Z
      COMMON    /MOD/AR,AC
C
      DATA FUNC(1)/1.0/
C
C* DATA STATEMENTS NEEDED FOR ARRAY REDIMENSIONING *******************
C
      DATA LCMOD,LRMOD/14,16/,        LSA,LSJ     /100,160/
```

```
      DATA LPT         /20/,                 LRDOF,LCDOF/115,50/
C
C***********************************************************************
C
      LDYN=2*(LRMOD+LCMOD)
      LG=MAX0(LRDOF,LCDOF)
C
C-----OPEN OUTPUT FILE
C     NOTES: FILE FIN.BAT IS EMPTY IF OUTPUT IS BEING LISTED ON PRN,
C            ELSE 'LIST2' IS READ INTO TAPE4.
C            CHANNEL 4 IS USED FOR PRINTING THE ROOT-LOCUS PLOT.
C
      OPEN(2,FILE='FIN.BAT',STATUS='OLD')
      TAPE4='PRN'
      READ(2,9000,END=5) TAPE4
    5 CLOSE(2)
      OPEN(6,FILE=TAPE4,STATUS='NEW')
C
C-----OPEN RUN DATA, EIGENVALUE, AND MODE SHAPE DATA FILES
C
      OPEN(1,FILE='RUNDATA.BIN',STATUS='OLD',FORM='UNFORMATTED')
      OPEN(2,FILE='EIGENS.BIN',STATUS='NEW',FORM='UNFORMATTED')
      OPEN(3,FILE='SHAPES.BIN',STATUS='NEW',FORM='UNFORMATTED')
C
C-----READ PROBLEM DATA
C
   10 READ(1)  (TITL(I,1),I=1,72),NCDOF,NCMOD,NRDOF,NRMOD,IPCNT
C
      READ(1)  ((AC(I,J),I=1,NCDOF),J=1,NCMOD),((AR(I,J),I=1,NRDOF),
     +         J=1,NRMOD),((GBAR(I,J),I=1,NRMOD),J=1,NRMOD),(WC(I),
     +         I=1,NCMOD),(WR(I),I=1,NRMOD)
C
C-----READ SUBCASE DATA
C
   20 READ(1)  (TITL(I,2),I=1,72),NSA,NSL,IPRT,KCRT,NXTC,NFGEN,MAXPTS,
     +         RPM1,DRPM,NI,NROT,NCASE,FHIGH,FLOW,RZ,IPLTF,IPRT2,RZ1,
     +         NZ,IPRT3,KRPM,NSTAT,IFLG,IFLG1,THETA,SCALE,ISCNT
C
      NSJ=4*NSL+NSA(1)+NSA(2)+NSA(3)+NSA(4)
      NNSA=4*NSL+MAX0(NSA(1),NSA(2),NSA(3),NSA(4))
C
      READ(1)  (ZETAC(I),I=1,NCMOD),(NPT(J),(FG(I,J),SPEED(I,J),
     +         I=1,MAXPTS),J=1,NFGEN),((JFUN(I,J),I=1,NSJ),J=1,2),
     +         ((S(I,J),DMP(I,J),(NRC(I,K,J),K=1,2),I=1,NNSA),J=1,4),
     +         (IROT(I),I=1,NROT),(ICASE(I),I=1,NCASE),(ZRPM(I),
     +         I=1,KRPM),(X(I),I=1,NSTAT)
C
C-----SAVE CRITICAL SPEED AND STABILITY PLOT DATA
C
      WRITE(2)  IPLTF,IPRT2,TITL,RZ,RZ1,RPM1,DRPM,NI,FHIGH,FLOW,
     +          NRMOD,NCMOD,NZ,KCRT,IPCNT,ISCNT,NXTC
C
C-----SAVE MODE SHAPE PLOTTING DATA
C
      WRITE(3)  IPRT,IPRT3,NCDOF,NCMOD,NRDOF,NRMOD,NCASE,NROT,TITL,
     +          NSTAT,IFLG,IFLG1,THETA,SCALE,IPCNT,ISCNT,NXTC
C
      WRITE(3)  ((AC(I,J),I=1,NCDOF),J=1,NCMOD),((AR(I,J),I=1,NRDOF),
     +          J=1,NRMOD),(ICASE(I),I=1,NCASE),(IROT(I),I=1,NROT),
```

```
     +          (X(I),I=1,NSTAT)
C
C-----F O R T O 3
C
      WRITE(6,9001) IPCNT,ISCNT
      WRITE(*,9001) IPCNT,ISCNT
C
      CALL FORT03(A,AC,AR,DMP,FG,GBAR,JFUN,NPT,NRC,W,S,SPEED,WC,WR,
     +            ZETAC,FUNC(2),LCDOF,LRDOF,LRMOD,LDYN,LSA,LSJ,LPT,
     +            LG,ICASE,IROT,WK,G,Z,ZRPM)
C
C-----RERUN?
C
      GOTO(10,20,20) NXTC
C
C-----CLOSE DATA FILES
C
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
C
C-----FORMAT STATEMENTS
C
 9000 FORMAT(/4X,A)
 9001 FORMAT(1H1///10X,'* * *   M A I N   R O U T I N E   * * *'//15X,
     +        'MODAL INPUT ',I2,5X,'SUBCASE ',I2///)
C
      END
```

Name	Type	Offset	P	Class
A	REAL*8	13370		
AC	REAL	7360		/MOD /
AR	REAL	0		/MOD /
DMP	REAL	9750		
DRPM	REAL	178		/DATA /
FG	REAL	11350		
FHIGH	REAL	184		/DATA /
FLOW	REAL	188		/DATA /
FUNC	REAL	13270		
G	REAL	0		/MEM /
GBAR	REAL*8	7702		
I	INTEGER*2	42188		
ICASE	INTEGER*2	7662		
IFLG	INTEGER*2	42218		
IFLG1	INTEGER*2	42220		
IPCNT	INTEGER*2	42192		
IPLTF	INTEGER*2	192		/DATA /
IPRT	INTEGER*2	168		/DATA /
IPRT2	INTEGER*2	194		/DATA /
IPRT3	INTEGER*2	196		/DATA /
IROT	INTEGER*2	7622		
ISCNT	INTEGER*2	42230		
J	INTEGER*2	42194		
JFUN	INTEGER*2	5334		
K	INTEGER*2	42236		
KCRT	INTEGER*2	170		/DATA /
KRPM	INTEGER*2	154		/DATA /
LCDOF	INTEGER*2	42182		

```
D Line# 1      7
LCMOD   INTEGER*2    42170
LDYN    INTEGER*2    42184
LG      INTEGER*2    42186
LPT     INTEGER*2    42178
LRDOF   INTEGER*2    42180
LRMOD   INTEGER*2    42172
LSA     INTEGER*2    42174
LSJ     INTEGER*2    42176
MAXO                        INTRINSIC
MAXPTS  INTEGER*2    42210
NCASE   INTEGER*2    42214
NCDOF   INTEGER*2        6  /DATA   /
NCMOD   INTEGER*2        2  /DATA   /
NFGEN   INTEGER*2      156  /DATA   /
NGYRO   INTEGER*2        0  /DATA   /
NI      INTEGER*2      182  /DATA   /
NNSA    INTEGER*2    42234
NPT     INTEGER*2     5974
NRC     INTEGER*2     6022
NRDOF   INTEGER*2        8  /DATA   /
NRMOD   INTEGER*2        4  /DATA   /
NROT    INTEGER*2    42212
NSA     INTEGER*2      158  /DATA   /
NSJ     INTEGER*2    42232
NSL     INTEGER*2      166  /DATA   /
NSTAT   INTEGER*2    42216
NXTC    INTEGER*2      172  /DATA   /
NZ      INTEGER*2     1758
RPM1    REAL           174  /DATA   /
RZ      REAL          1766
RZ1     REAL          1790
S       REAL          3734
SCALE   REAL         42226
SPEED   REAL          1814
TAPE4   CHAR*14         10  /DATA   /
THETA   REAL         42222
TITL    CHAR*1          10  /DATA   /
W       REAL          1278
WC      REAL           158
WK      REAL*8         318
WR      REAL           214
X       REAL           278
Z       REAL*8           0  /MEM    /
ZETAC   REAL             2
ZRPM    REAL            58


Name    Type          Size  Class

DATA                   198  COMMON
FORT03                      SUBROUTINE
MEM                  57600  COMMON
MOD                  10160  COMMON
RSTAB                       PROGRAM

  Pass One     No Errors Detected
              146 Source Lines
```

D Line# 1 7 Microsoft FORTRAN77 V3.13 8/05/83

```
*FORT03.FOR*****************************************************************
C
C     S U B R O U T I N E   F O R T 0 3
C
C***************************************************************************
C
C     SUBROUTINES UTILIZED IN FORT03 ARE CALLED
C     IN THE FOLLOWING ORDER:
C
C     MATX . . . . . . . . . . . FORMS THE MATRIX ADDITIONS INTO THE
C                                NORMALIZED DYNAMICAL MATRIX "A"
C
C     EIGRF. . . . . . . . . . . FROM THE IMSL LIBRARY. SOLVES A GENERAL
C                                MATRIX FOR COMPLEX ROOTS AND EIGENVECTORS
C
C     ANSR . . . . . . . . . . . SAVES ITERATION STEP RESULTS AND PLOTS
C                                EIGENVALUES ON CRT (ROOT LOCUS)
C
C***************************************************************************
C
$STORAGE:2
C
      SUBROUTINE FORT03(A,AC,AR,DMP,FG,GBAR,JFUN,NPT,NRC,W,S,SPEED,WC,
     +                  WR,ZETAC,FUNC,LCDOF,LRDOF,LRMOD,LDYN,LSA,LSJ,
     +                  LPT,LG,ICASE,IROT,WK,G,Z,ZRPM)
C
      CHARACTER TITL(72,2)*1
C
      REAL*8    A(LDYN,1),WK(1),GBAR(LRMOD,1),Z(2,LDYN,1)
C
      INTEGER   JFUN(LSJ,1),NPT(1),NRC(LSA,2,1),ICASE(1),IROT(1),NZ(4),
     +          NSA(4)
C
      DIMENSION AC(LCDOF,1),AR(LRDOF,1),WC(1),WR(1),S(LSA,1),G(LG,1),
     +          DMP(LSA,1),FG(LPT,1),SPEED(LPT,1),ZETAC(1),FUNC(1),
     +          ZRPM(1),W(2,1)
C
      COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
     +               NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
     +               IFLTF,IPRT2,IPRT3
C
C-----M A I N   R O U T I N E
C
      NMODES=NRMOD+NCMOD
      DO 200 IRPM = 1,NI+1
        RPM  =  RPM1 + (IRPM - 1) * DRPM
        WRITE(6,36) TITL,RPM
        WRITE(*,36) TITL,RPM
C
C-------UPDATE FUNCTION GENERATORS
C
        DO 100 I=1,NFGEN
        DO 10 J=1,NPT(I)-1
   10   IF(RPM.LT.SPEED(J,I)) GOTO 100
  100   FUNC(I)=FG(J-1,I)+(FG(J,I)-FG(J-1,I))*(RPM-SPEED(J-1,I))/
     +          (SPEED(J,I)-SPEED(J-1,I))
C
C-------ZERO THE DYNAMICAL MATRIX
C
```

```
D Line# 1      7
1                 DO 400 I=1,2*NMODES
2                   DO 400 J=1,2*NMODES
3          400    A(I,J)=0.0
1             C
1             C-------SET THE IDENTITY MASS PARTITION OF A
1             C
1                 DO 300 I = 1,NMODES
2          300    A(I+NMODES,I) = 1.0
1             C
1             C-------LOAD NSL'S AND NSA'S INTO A
1             C
1                 CALL MATX(AC,AR,DMP,JFUN,LCDOF,LRDOF,LSA,LSJ,LDYN,NRC,S,
1                +          A,G,LG,WK,FUNC)
1             C
1             C-------ADD SPECTRAL MATRIX, DAMPING MATRIX, AND GYROSCOPICS TO A
1             C
1                 DO 410     I = 1,  NRMOD
2                   A(I,NMODES+I) = A(I,NMODES+I) - WR(I)*WR(I)
2                   DO 410     J = 1,  NRMOD
3          410    A(I,J) = A(I,J) - GBAR(I,J)*RPM*0.1047197
1             C
1                 DO 430     I = 1,  NCMOD
2                   IR  =  NRMOD  +  I
2                   A(IR,IR)  =  A(IR,IR)  -  2.*ZETAC(I)*WC(I)
2          430    A(IR,NMODES+IR) = A(IR,NMODES+IR) - WC(I)*WC(I)
1             C
1             C-------CALCULATE EIGENVALUES AND EIGENVECTORS
1             C
1                 CALL EIGRF(A,2*NMODES,LDYN,W,Z,LDYN,WK,IER)
1             C
1             C-------STORE RESULTS
1             C
1                 CALL ANSR(Z,LDYN,RPM,W,ZRPM)
1             C
1          200 CONTINUE
              C
              C-----WRITE END-OF-CASE RECORD ON EIGENVECTOR FILE
              C
                   WRITE(3) RPM,-9999,(W(J,1),J=1,2),
                  +                    ((SNGL(Z(J,L,1)),J=1,2),L=1,NMODES)
              C
              C-----FORMAT STATEMENTS
              C
                36 FORMAT(1H1/5X,72A//5X,72A///5X,'ROTOR SPIN SPEED =',F8.0,'  RPM')
              C
                   END
```

Name	Type	Offset	P	Class
A	REAL*8	0	*	
AC	REAL	4	*	
AR	REAL	8	*	
DMP	REAL	12	*	
DRPM	REAL	178		/DATA /
FG	REAL	16	*	
FHIGH	REAL	184		/DATA /
FLOW	REAL	188		/DATA /
FUNC	REAL	60	*	
G	REAL	108	*	

```
GBAR     REAL*8          20 *
I        INTEGER*2       24
ICASE    INTEGER*2       96 *
IER      INTEGER*2       78
IPLTF    INTEGER*2      192    /DATA   /
IPRT     INTEGER*2      168    /DATA   /
IPRT2    INTEGER*2      194    /DATA   /
IPRT3    INTEGER*2      196    /DATA   /
IR       INTEGER*2       76
IROT     INTEGER*2      100 *
IRPM     INTEGER*2       12
J        INTEGER*2       32
JFUN     INTEGER*2       24 *
KCRT     INTEGER*2      170    /DATA   /
KRPM     INTEGER*2      154    /DATA   /
L        INTEGER*2       82
LCDOF    INTEGER*2       64 *
LDYN     INTEGER*2       76 *
LG       INTEGER*2       92 *
LPT      INTEGER*2       88 *
LRDOF    INTEGER*2       68 *
LRMOD    INTEGER*2       72 *
LSA      INTEGER*2       80 *
LSJ      INTEGER*2       84 *
NCDOF    INTEGER*2        6    /DATA   /
NCMOD    INTEGER*2        2    /DATA   /
NFGEN    INTEGER*2      156    /DATA   /
NGYRO    INTEGER*2        0    /DATA   /
NI       INTEGER*2      182    /DATA   /
NMODES   INTEGER*2       10
NPT      INTEGER*2       28 *
NRC      INTEGER*2       32 *
NRDOF    INTEGER*2        8    /DATA   /
NRMOD    INTEGER*2        4    /DATA   /
NSA      INTEGER*2      158    /DATA   /
NSL      INTEGER*2      166    /DATA   /
NXTC     INTEGER*2      172    /DATA   /
NZ       INTEGER*2        2
RPM      REAL            20
RPM1     REAL           174    /DATA   /
S        REAL            40 *
SNGL                           INTRINSIC
SPEED    REAL            44 *
TITL     CHAR*1          10    /DATA   /
W        REAL            36 *
WC       REAL            48 *
WK       REAL*8         104 *
WR       REAL            52 *
Z        REAL*8         112 *
ZETAC    REAL            56 *
ZRPM     REAL           116 *

     106 *MATX.FOR*******************************************************************
     107 C
     108 C          MATX  COORDINATES THE INTRAGROUP, INTERGROUP, MATRIX
     109 C                ADDITIONS AND THE CALCULATION OF THEIR STIFFNESS
     110 C                AND DAMPING VALUES (MULTIPLIED BY THE RESPECTIVE
     111 C                FUNCTION GENERATOR).
```

```
C
C*********************************************************************
C
      SUBROUTINE MATX(AC,AR,DMP,JFUN,LCDOF,LRDOF,LSA,LSJ,LDYN,NRC,S,
     +                A,G,LG,WK,FUNC)
C
      INTEGER     JFUN(LSJ,1),NRC(LSA,2,1),NSA(4)
C
      DIMENSION   AR(LRDOF,1),AC(LCDOF,1),DMP(LSA,1),S(LSA,1),
     +            G(LG,1),FUNC(1)
C
      REAL*8      A(LDYN,1),WK(1)
C
      CHARACTER TITL(72,2)*1
C
      COMMON     /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
     +                 NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
     +                 IPLTF,IPRT2,IPRT3
C
C-----ZERO THE G MATRIX
C
      DO 100      I = 1,LG
      DO 100      J = 1,LG
  100 G(J,I)   = 0.0
C
C-----TRANSFORM MATRIX ADDITIONS INTO NORMAL COORDINATES,
C     AND ADD THEM TO THE SYSTEM DYNAMICAL MATRIX "A".
C
C     1) ROTOR-TO-ROTOR ADDITIONS
C     2) ROTOR-TO-CASING ADDITIONS
C     3) CASING-TO-ROTOR ADDITIONS
C     4) CASING-TO-CASING ADDITIONS
C
      N   =   4*NSL
      NMODES = NRMOD + NCMOD
      IK=0
C
      DO 1000 IP = 1,4
C
        IF ( NSL+NSA(IP) .EQ. 0 ) GOTO 1000
C
C-------ADD STIFFNESS ADDITIONS
C
        IF(NSL.NE.0) THEN
          DO 200 I = 1,N
  200     G(NRC(I,1,IP),NRC(I,2,IP))   =  S(I,IP)*FUNC(JFUN(I,1))   +
     1                                    G(NRC(I,1,IP),NRC(I,2,IP))
        ENDIF
C
        IF(NSA(IP).NE.0) THEN
          DO 250 I   =   1,NSA(IP)
          NN=N+I
  250     G(NRC(NN,1,IP),NRC(NN,2,IP))=S(NN,IP)*FUNC(JFUN(NN+IK,1))+
     1                                G(NRC(NN,1,IP),NRC(NN,2,IP))
        ENDIF
C
        IF(IP.EQ.1) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AR,LRDOF,
     1                     NRDOF,NRMOD,A(1,NMODES+1),LDYN,WK)
        IF(IP.EQ.2) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AC,LCDOF,
```

```
D Line# 1      7                                   Microsoft FORTRAN77 V3.13 8/05/83
1               1                         NCDOF,NCMOD,A(1,NMODES+NRMOD+1),LDYN,WK)
1                   IF(IP.EQ.3) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AR,LRDOF,
1               1                         NRDOF,NRMOD,A(NRMOD+1,NMODES+1),LDYN,WK)
1                   IF(IP.EQ.4) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AC,LCDOF,
1               1                         NCDOF,NCMOD,A(NRMOD+1,NMODES+NRMOD+1),LDYN,WK)
1         C
1         C-------RESET G
1         C
1                   DO 300     I = 1, NSA(IP)+N
2           300     G(NRC(I,1,IP),NRC(I,2,IP)) = 0.0
1         C
1         C-------ADD DAMPING MODIFICATIONS
1         C
1                   IF(NSL.NE.0) THEN
1                     DO 350  I  =  1,N
2           350       G(NRC(I,1,IP),NRC(I,2,IP))  =  DMP(I,IP)*FUNC(JFUN(I,2))  +
2               1                                    G(NRC(I,1,IP),NRC(I,2,IP))
1                   ENDIF
1         C
1                   IF(NSA(IP).NE.0) THEN
1                     DO 400 I  =  1,NSA(IP)
2                     NN=N+I
2           400       G(NRC(NN,1,IP),NRC(NN,2,IP))=DMP(NN,IP)*FUNC(JFUN(NN+IK,2))+
2               1                                  G(NRC(NN,1,IP),NRC(NN,2,IP))
1                   ENDIF
1         C
1                   IF(IP.EQ.1) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AR,LRDOF,
1               1                              NRDOF,NRMOD,A,LDYN,WK)
1                   IF(IP.EQ.2) CALL TRANS(AR,LRDOF,NRDOF,NRMOD,G,LG,AC,LCDOF,
1               1                             NCDOF,NCMOD,A(1,NRMOD+1),LDYN,WK)
1                   IF(IP.EQ.3) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AR,LRDOF,
1               1                              NRDOF,NRMOD,A(NRMOD+1,1),LDYN,WK)
1                   IF(IP.EQ.4) CALL TRANS(AC,LCDOF,NCDOF,NCMOD,G,LG,AC,LCDOF,
1               1                              NCDOF,NCMOD,A(NRMOD+1,NRMOD+1),LDYN,WK)
1         C
1         C-------RESET G
1         C
1                   DO 450     I = 1, NSA(IP)+N
2           450     G(NRC(I,1,IP),NRC(I,2,IP)) = 0.0
1         C
1        1000 IK = IK + NSA(IP)
          C
                  END
```

```
Name    Type          Offset P Class

A       REAL*8           44 *
AC      REAL              0 *
AR      REAL              4 *
DMP     REAL              8 *
DRPM    REAL            178   /DATA  /
FHIGH   REAL            184   /DATA  /
FLOW    REAL            188   /DATA  /
FUNC    REAL             60 *
G       REAL             48 *
I       INTEGER*2       148
IK      INTEGER*2       168
IP      INTEGER*2       170
IPLTF   INTEGER*2       192   /DATA  /
```

```
D Line# 1      7
IPRT    INTEGER*2       168   /DATA   /
IPRT2   INTEGER*2       194   /DATA   /
IPRT3   INTEGER*2       196   /DATA   /
J       INTEGER*2       156
JFUN    INTEGER*2        12 *
KCRT    INTEGER*2       170   /DATA   /
KRPM    INTEGER*2       154   /DATA   /
LCDOF   INTEGER*2        16 *
LDYN    INTEGER*2        32 *
LG      INTEGER*2        52 *
LRDOF   INTEGER*2        20 *
LSA     INTEGER*2        24 *
LSJ     INTEGER*2        28 *
N       INTEGER*2       164
NCDOF   INTEGER*2         6   /DATA   /
NCMOD   INTEGER*2         2   /DATA   /
NFGEN   INTEGER*2       156   /DATA   /
NGYRO   INTEGER*2         0   /DATA   /
NI      INTEGER*2       182   /DATA   /
NMODES  INTEGER*2       166
NN      INTEGER*2       184
NRC     INTEGER*2        36 *
NRDOF   INTEGER*2         8   /DATA   /
NRMOD   INTEGER*2         4   /DATA   /
NSA     INTEGER*2       158   /DATA   /
NSL     INTEGER*2       166   /DATA   /
NXTC    INTEGER*2       172   /DATA   /
RPM1    REAL            174   /DATA   /
S       REAL             40 *
TITL    CHAR*1           10   /DATA   /
WK      REAL*8           56 *
```

```
*ANSR.FOR*****************************************************************
C
C      ANSR ORGANIZES OUTPUT INFORMATION AND PLOTS,SAVES
C      THAT INFORMATION PER USER REQUEST.
C      ANSR CALLS SUBROUTINE ZSORT.
C
C*************************************************************************
C
       SUBROUTINE ANSR(Z,LDYN,RPM,W,ZRPM)
C
       REAL*8     Z(2,LDYN,1),WZ
C
       DIMENSION  ZRPM(1),W(2,1),NSA(4)
C
       LOGICAL    PLOT
C
       CHARACTER TITL(72,2)*1,TAPE4*10
C
       COMMON    /DATA/NGYRO,NCMOD,NRMOD,NCDOF,NRDOF,TITL,KRPM,NFGEN,
      +                NSA,NSL,IPRT,KCRT,NXTC,RPM1,DRPM,NI,FHIGH,FLOW,
      +                IPLTF,IPRT2,IPRT3
C
C-----COMPRESS THE EIGENVALUE AND EIGENVECTOR ARRAYS
C      (CORRECTING FOR REAL ROOTS)
C
       N  =  NRMOD + NCMOD
```

```
        K=1
C
        DO 20  I = 1,N
1        IFIX=I*2-K
1        IF(W(2,IFIX).EQ.O.) THEN
1         IF(K.EQ.1) THEN
1          K=2
1         ELSE
1          K=1
1          IFIX=I*2-K
1         ENDIF
1        ENDIF
1       W(1,I)=W(1,IFIX)
1       W(2,I)=W(2,IFIX)
1       DO 20 J=1,N
2        Z(1,J,I)=Z(1,J+N,IFIX)
2    20  Z(2,J,I)=Z(2,J+N,IFIX)
C
C-----SORT THE ARRAYS WITH RESPECT TO FREQUENCY
C
        N2=N
C
        DO 9 I=1,N-1
1         N2=N2-1
1         DO 9 J=1,N2
2           IF(W(2,J).LE.W(2,J+1)) GOTO 9
2           WZ=W(1,J+1)
2           W(1,J+1)=W(1,J)
2           W(1,J)=WZ
2           WZ=W(2,J+1)
2           W(2,J+1)=W(2,J)
2           W(2,J)=WZ
2           DO 8 K=1,N
3             WZ=Z(1,K,J+1)
3             Z(1,K,J+1)=Z(1,K,J)
3             Z(1,K,J)=WZ
3             WZ=Z(2,K,J+1)
3             Z(2,K,J+1)=Z(2,K,J)
3     8       Z(2,K,J)=WZ
2     9 CONTINUE
C
C-----DETERMINE WHETHER THE MODE SHAPES SHOULD BE STORED FOR
C       PRINTING/PLOTTING LATER
C
        PLOT=.FALSE.
        IF(IPRT3.EQ.O.OR.IPRT.EQ.O) THEN
          J=1
   24    IF(J.LE.KRPM) THEN
            IF(RPM.EQ.ZRPM(J)) PLOT=.TRUE.
            J=J+1
            GOTO 24
          ENDIF
        ENDIF
C
C-----STORE AND PLOT RESULTS
C
        WRITE(6,7)
        WRITE(*,7)
C.....FOR EACH MODE, WRITE RESULTS
```

```
D Line# 1       7                                     Microsoft FORTRAN77 V3.13 8/05/83
                DO 35 I=1,N
1                 C = W(2,I) * 9.54929
1                 WRITE(6,30) I,(W(J,I),J=1,2),C
1                 WRITE(*,30) I,(W(J,I),J=1,2),C
1                 IF(.NOT.PLOT) GOTO 35
1           C.......IF THIS MODE SHAPE IS TO BE PLOTTED, STORE SHAPE
1                 IF(C.GE.FLOW.AND.C.LE.FHIGH) WRITE(3) RPM,I,(W(J,I),J=1,2),
1              +                                ((SNGL(Z(J,L,I)),J=1,2),L=1,N)
1             35 CONTINUE
            C.....STORE EIGENVALUES
                WRITE(2) ((W(J,I),J=1,2),I=1,N)
            C
            C-----FORMAT STATEMENTS
            C
               7 FORMAT(//5X,'MODE',11X,'REAL',14X,'IMAGINARY',14X,'CPM'/)
              30 FORMAT(4X,I4,3(5X,1PE15.5))
            8000 FORMAT(4HRPM=,F6.0)
            C
                END
```

Name	Type	Offset	P	Class
A	REAL	0	*	
C	REAL	24	*	
D	REAL*8	40	*	
G	REAL	16	*	
I	INTEGER*2	414		
J	INTEGER*2	422		
K	INTEGER*2	430		
L	INTEGER*2	438		
LRA	INTEGER*2	4	*	
LRC	INTEGER*2	28	*	
LRD	INTEGER*2	44	*	
LRG	INTEGER*2	20	*	
M	INTEGER*2	446		
NCA	INTEGER*2	12	*	
NCC	INTEGER*2	36	*	
NRA	INTEGER*2	8	*	
NRC	INTEGER*2	32	*	
WK	REAL*8	48	*	

```
    378 $LIST
```

Name	Type	Size	Class
ANSR			SUBROUTINE
DATA		198	COMMON
EIGRF			SUBROUTINE
FORT03			SUBROUTINE
MATX			SUBROUTINE
TRANS			SUBROUTINE

```
 Pass One     No Errors Detected
              378 Source Lines
```

```
*PSTRSTAB.FOR*****************************************************************
C
C      P R O G R A M   P S T R S T A B
C
C      ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C*****************************************************************************
C
C      PRINCIPAL SUBROUTINES UTILIZED IN PSTRSTAB ARE CALLED
C      IN THE FOLLOWING ORDER:
C
C      FNCPLT . . . . . . . . . FUNCTION GENERATOR PLOTS
C
C      CRTPLT . . . . . . . . . CRITICAL SPEED PLOTS
C
C      STBPLT . . . . . . . . . STABILITY PLOTS
C
C      SHPPLT . . . . . . . . . MODE SHAPE PLOTS
C
C*****************************************************************************
C
C      INPUT FILE NAMES    DESCRIPTION
C      ----------------    ------------------------------------------------
C
C      FGPLTS.BIN          BINARY FILE CREATED BY PROGRAM PRERSTAB
C                          CONTAINING THE FUCTION GENERATOR DATA FOR
C                          PLOTTING
C
C      EIGENS.BIN          BINARY FILE CREATED BY PROGRAM RSTAB
C                          CONTAINING THE EIGENVALUES AT EACH RPM STEP
C                          FOR CRITICAL SPEED AND STABILITY PLOTTING
C
C      SHAPES.BIN          BINARY FILE CREATED BY PROGRAM RSTAB
C                          CONTAINING THE COMPLEX MODE SHAPES FOR MODE
C                          SHAPE PLOTTING
C
C      FIN.BAT             FORMATTED IBM BATCH FILE USED TO DETERMINE
C                          WHETHER TO PRINT THE OUTPUT LISTING DATA
C                          IMMEDIATELY, OR TO WRITE THE INFORMATION TO
C                          TEMPORARY FILE 'LIST3'
C
C      OUTPUT FILE NAMES   DESCRIPTION
C      -----------------   ------------------------------------------------
C
C      LIST3               FORMATTED OUTPUT LISTING CREATED ONLY IF
C                          OUTPUT IS NOT PRINTED IMMEDIATELY
C
C      VECT.TMP,MAPS.TMP   BINARY TEMPORARY FILES CREATED BY THE PLOTT88
C                          LIBRARY. THESE WILL BE DELETED UPON NORMAL
C                          TERMINATION OF THE PLOTT88 ROUTINES
C
C*****************************************************************************
$STORAGE:2
C
       PROGRAM PSTRSTAB
C
       CHARACTER TITL(72,2)*1,TYTL(48)*1,BANNR(3)*31,TAPE4*14
C
       EQUIVALENCE (AR,W),(AR,FG),(TITL,TAPE4),(AC,SP)
```

```
C
C* ARRAY DIMENSIONS MOST LIKELY TO CHANGE ****************************
C
C-----NOTE: DIMENSION VECT     2, NROT+NCASE
C                     W        2, LMOD , NI+1
C                     Z        2, LMOD
C                     FG,SP    NPTS+2
C
      DIMENSION AR(115,16),AC(50,14),FG(22),SP(22),ICASE(20),IROT(20),
     +          X(10),VECT(2,40),W(2,30,51),Z(2,30)
C
C*******************************************************************
C
      COMMON    /MEM/AR,AC,VECT,Z,ICASE,IROT
C
      COMMON    /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,RZ(6),IPLTF,
     +                IPRT2,IPRT3,RZ1(6),NZ(4),NCASE,NROT,NSTAT,IFLG,
     +                IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,NI,IPCNT,ISCNT
C
      COMMON    /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),
     +                VECTY(4)
C
C* DATA STATEMENTS NEEDED FOR ARRAY REDIMENSIONING ******************
C
      DATA LCDOF,LRDOF,LMOD/50,115,30/
C
C********************************************************************
C
      DIMX=6.1125
      DIMY=4.8900
      XASP=.890
C
C-----OPEN LISTING FILE
C
      OPEN(2,FILE='FIN.BAT',STATUS='OLD')
      TAPE4='PRN'
      READ(2,9000,END=5) TAPE4
    5 CLOSE(2)
      OPEN(6,FILE=TAPE4,STATUS='NEW')
C
C-----OPEN PLOT DATA FILES
C
      OPEN(1,FILE='FGPLTS.BIN',STATUS='OLD',FORM='UNFORMATTED')
      OPEN(2,FILE='EIGENS.BIN',STATUS='OLD',FORM='UNFORMATTED')
      OPEN(3,FILE='SHAPES.BIN',STATUS='OLD',FORM='UNFORMATTED')
C
C-----INITIALIZE CRT
C
      CALL QBORD(1)
      CALL QCSIZ(0,0)
C
C-----INITIATE PLOT88 USAGE
C
   10 CALL PLOTS(0,0,1)
C
C-----READ SUBCASE DATA
C
      READ(2) IPLTF,IPRT2,TITL,RZ,RZ1,RPM1,DRPM,NI,FHIGH,FLOW,NRMOD,
     +        NCMOD,NZ,KCRT,IPCNT,ISCNT,NXTC
```

```
C
      READ(3) IPRT,IPRT3,NCDOF,NCMOD,NRDOF,NRMOD,NCASE,NROT,TITL,
     +        NSTAT,IFLG,IFLG1,TH,SC,IPCNT,ISCNT,NXTC
C
C-----SUBCASE HEADER
C
      CALL QCLEAR(1,7)
      CALL QCMOV(0,24)
C
      WRITE(6,8002) IPCNT,ISCNT,TITL
      WRITE(*,8002) IPCNT,ISCNT,TITL
C
      WRITE(6,8003) NRDOF,NRMOD,NCDOF,NCMOD
      WRITE(*,8003) NRDOF,NRMOD,NCDOF,NCMOD
C
      WRITE(6,8004) RPM1,DRPM,NI
      WRITE(*,8004) RPM1,DRPM,NI
C
C-----F U N C T I O N   G E N E R A T O R S
C
      READ(1) IPCNT,ISCNT
C
      FNY=10.
      FNX=ANINT(FNY*1.25)
C
      FACTY=DIMY/FNY
      FACTX=DIMX/FNX
C
   15 READ(1) IFGEN,TYTL,TITL,NPT,(SF(I),FG(I),I=1,NPT)
      IF(IFGEN.GT.0) THEN
        CALL FNCPLT(FG,SF,NPT)
        GOTO 15
      ENDIF
C
C-----C R I T I C A L   S P E E D S / S T A B I L I T Y
C
      N = NCMOD + NRMOD
C
      DO 20 J=1,NI+1
        READ(2) ((W(K,I,J),K=1,2),I=1,N)
   20 CONTINUE
      IF(IPLTF.EQ.0.OR.IPRT2.EQ.0) THEN
        CALL CRTPLT(W,LMOD,MHIGH)
        IF(IPRT2.EQ.0.AND.MHIGH.NE.0) CALL STBPLT(W,LMOD,MHIGH)
      ENDIF
C
C-----R O O T   L O C U S      (TO BE WRITTEN)
C
C     CALL LOCUS(W,LMOD)
C
C-----M O D E   S H A P E S
C
      READ(3) ((AC(I,J),I=1,NCDOF),J=1,NCMOD),((AR(I,J),I=1,NRDOF),
     +        J=1,NRMOD),(ICASE(I),I=1,NCASE),(IROT(I),I=1,NROT),
     +        (X(I),I=1,NSTAT)
C
      CALL SHPPLT(VECT,Z,AC,AR,ICASE,IROT,LMOD,LCDOF,LRDOF,X)
C
C-----DUMP PLOT BUFFER
```

```
C
      CALL PLOT(0.,0.,999)
C
C-----RERUN?
C
      IF(NXTC.NE.4) GOTO 10
C
C-----CLOSE DATA FILES
C
      CLOSE(1)
      CLOSE(2)
      CLOSE(3)
C
C-----COMPLETION BANNER
C
      CALL QTIME(NZ,NZ(2),NZ(3),NZ(4))
      NZ(3) = NZ(3) + NZ(4)/100. + 0.5
      WRITE(6,8000) (NZ(I),I=1,3)
      WRITE(*,8000) (NZ(I),I=1,3)
C
C-----CLOSE LISTING FILE
C
      CLOSE(6)
C
C-----FORMAT STATEMENTS
C
 8000 FORMAT(6(/),2X,78(1H*)/7H  *****,68X,5H*****/7H  *****,18X,
     +      22HEXECUTION COMPLETED AT,I3,2(1H:,I2),19X,5H*****/
     +      7H  *****,68X,5H*****/2X,78(1H*))
 8002 FORMAT(1H1///5X,4('*  '),' S U B C A S E   P L O T T I N G ',
     +       4('  *')///20X,'MODAL INPUT',I2,5X,'SUBCASE',I2/2(//5X,72A)
     +       //)
 8003 FORMAT(5X,'NO. OF ROTOR DOF  =',I4,5X,'NO. OF ROTOR MODES  =',I4//
     +       5X,'NO. OF CASING DOF =',I4,5X,'NO. OF CASING MODES =',I4/)
 8004 FORMAT(/5X,'RPM1  =',1PE13.5,8X,'DRPM  =',E13.5,8X,'IP  =',I4)
 9000 FORMAT(//4X,A)
C
      END
```

Name	Type	Offset	P	Class
AC	REAL	7360		/MEM /
ANINT				INTRINSIC
AR	REAL	0		/MEM /
BANNR	CHAR*31	42		
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FG	REAL	0		/MEM /
FHIGH	REAL	142		
FLOW	REAL	146		
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
I	INTEGER*2	156		
ICASE	INTEGER*2	10720		/MEM /
IFGEN	INTEGER*2	152		
IFLG	INTEGER*2	268		/DATA /

```
IFLG1    INTEGER*2      270    /DATA    /
IPCNT    INTEGER*2      294    /DATA    /
IPLTF    INTEGER*2      224    /DATA    /
IPRT     INTEGER*2      280    /DATA    /
IPRT2    INTEGER*2      226    /DATA    /
IPRT3    INTEGER*2      228    /DATA    /
IROT     INTEGER*2    10760    /MEM     /
ISCNT    INTEGER*2      296    /DATA    /
J        INTEGER*2      162
K        INTEGER*2      170
KCRT     INTEGER*2      282    /DATA    /
LCDOF    INTEGER*2      136
LMOD     INTEGER*2      140
LRDOF    INTEGER*2      138
MHIGH    INTEGER*2      174
N        INTEGER*2      160
NCASE    INTEGER*2      262    /DATA    /
NCDOF    INTEGER*2        4    /DATA    /
NCMOD    INTEGER*2        0    /DATA    /
NI       INTEGER*2      292    /DATA    /
NPT      INTEGER*2      154
NRDOF    INTEGER*2        6    /DATA    /
NRMOD    INTEGER*2        2    /DATA    /
NROT     INTEGER*2      264    /DATA    /
NSTAT    INTEGER*2      266    /DATA    /
NXTC     INTEGER*2      150
NZ       INTEGER*2      254    /DATA    /
RPM1     REAL           284    /DATA    /
RZ       REAL           200    /DATA    /
RZ1      REAL           230    /DATA    /
SC       REAL           276    /DATA    /
SP       REAL          7360    /MEM     /
TAPE4    CHAR*14          8    /DATA    /
TH       REAL           272    /DATA    /
TITL     CHAR*1           8    /DATA    /
TYTL     CHAR*1         152    /DATA    /
VECT     REAL         10160    /MEM     /
VECTX    REAL            28    /DIMS    /
VECTY    REAL            44    /DIMS    /
W        REAL             0    /MEM     /
X        REAL             2
XASP     REAL            24    /DIMS    /
Z        REAL         10480    /MEM     /


Name     Type          Size    Class

CRTPLT                         SUBROUTINE
DATA                    298    COMMON
DIMS                     60    COMMON
FNCPLT                         SUBROUTINE
MEM                   12240    COMMON
PLOT                           SUBROUTINE
PLOTS                          SUBROUTINE
PSTRST                         PROGRAM
QBORD                          SUBROUTINE
QCLEAR                         SUBROUTINE
QCMOV                          SUBROUTINE
```

```
D Line# 1       7
QCSIZ                        SUBROUTINE
QTIME                        SUBROUTINE
SHFPLT                       SUBROUTINE
STBPLT                       SUBROUTINE

 Pass One     No Errors Detected
             215 Source Lines
```

```
D Line# 1     7                                    Microsoft FORTRAN77 V3.13 8/05/83
 *FNCPLT*******************************************************************
 C
 C      S U B R O U T I N E   F N C P L T
 C
 C      PLOTS FUNCTION GENERATORS
 C
 C*************************************************************************
 C
 $STORAGE:2
 C
        SUBROUTINE FNCPLT(FG,SP,N)
 C
        DIMENSION FG(1),SP(1)
 C
        CHARACTER*1 TITL(72,2),TYTL(48)
 C
        COMMON    /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,RZ(6),IPLTF,
      +                IPRT2,IPRT3,RZ1(6),NZ(4),NCASE,NROT,NSTAT,IFLG,
      +                IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,NI,IPCNT,ISCNT
 C
        COMMON    /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),
      +                VECTY(4)
 C
 C-----PLOT HEADING AND SET PLOT ORIGIN
 C
        CALL HEADIN(.193)
 C
 C-----SCALE FUNCTION GENERATOR
 C
        CALL SCALE(SP,FNY,N,1)
        CALL SCALE(FG,FNX,N,1)
 C
 C-----DRAW X/Y AXES
 C
        CALL AXES('AMPLITUDE',9,FG(N+1),FG(N+2),
      +           'SPEED  (RPM)',12,SP(N+1),SP(N+2))
 C
 C-----FIX X AXIS FOR PLOTTING ON 180. DEGREES
 C      AND DISTORTED SCALE
 C
        FG(N+2)=-FG(N+2)*FACTY/FACTX
 C
 C-----PLOT FUNCTION
 C
        CALL LINE(FG,SP,N,1,1,2)
 C
 C------PLOT BORDER
 C
        CALL NEWPEN(2)
        CALL BOX(0.,0.,-FNX*FACTX/FACTY,FNY)
        CALL NEWPEN(1)
 C
 C-----PLOT DASHED LINES IN X DIRECTION
 C
        CALL STDASH(.03760/FACTY,.03760/FACTY)
        DO 10 I=1,INT(FNY)
1         CALL PLOTD(0.,FLOAT(I),3)
1    10 CALL PLOTD(-DIMX/FACTY,FLOAT(I),2)
 C
```

```
C-----PLOT DASHED LINES IN Y DIRECTION
C
      DO 15 I=1,INT(FNX)
        CALL PLOTD(-I*FACTX/FACTY,0.,3)
   15 CALL PLOTD(-I*FACTX/FACTY,FNY,2)
C
C-----END OF PLOT
C
      CALL PLOT(0.,0.,-999)
C
      END
```

Name	Type	Offset	P	Class
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FG	REAL	0	*	
FLOAT				INTRINSIC
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
I	INTEGER*2	2		
IFLG	INTEGER*2	268		/DATA /
IFLG1	INTEGER*2	270		/DATA /
INT				INTRINSIC
IPCNT	INTEGER*2	294		/DATA /
IPLTF	INTEGER*2	224		/DATA /
IPRT	INTEGER*2	280		/DATA /
IPRT2	INTEGER*2	226		/DATA /
IPRT3	INTEGER*2	228		/DATA /
ISCNT	INTEGER*2	296		/DATA /
KCRT	INTEGER*2	282		/DATA /
N	INTEGER*2	8	*	
NCASE	INTEGER*2	262		/DATA /
NCDOF	INTEGER*2	4		/DATA /
NCMOD	INTEGER*2	0		/DATA /
NI	INTEGER*2	292		/DATA /
NRDOF	INTEGER*2	6		/DATA /
NRMOD	INTEGER*2	2		/DATA /
NROT	INTEGER*2	264		/DATA /
NSTAT	INTEGER*2	266		/DATA /
NZ	INTEGER*2	254		/DATA /
RPM1	REAL	284		/DATA /
RZ	REAL	200		/DATA /
RZ1	REAL	230		/DATA /
SC	REAL	276		/DATA /
SP	REAL	4	*	
TH	REAL	272		/DATA /
TITL	CHAR*1	8		/DATA /
TYTL	CHAR*1	152		/DATA /
VECTX	REAL	28		/DIMS /
VECTY	REAL	44		/DIMS /
XASP	REAL	24		/DIMS /

```
*CRTPLT.FOR************************************************************
C
```

```
C      S U B R O U T I N E   C R T P L T
C
C      CRTPLT PLOTS CRITICAL SPEEDS
C
C      NOTE: CRTPLT RETURNS MHIGH (HIGHEST MODE NUMBER FITTING INTO
C            THE SPECIFIED BOUNDS FOR THE CRITICAL SPEED PLOT). THIS
C            NUMBER IS NEEDED BY SUBROUTINE STBPLT, SO IT MUST BE
C            CALCULATED REGARDLESS OF WHETHER THE CRITICAL SPEED PLOT
C            WAS REQUESTED.
C
C***********************************************************************
C
       SUBROUTINE CRTPLT(W,LMOD,MHIGH)
C
       CHARACTER  TITL(72,2)*1,TYTL*48
C
       DIMENSION  W(2,LMOD,1),VECTX(4),VECTY(4)
C
       EQUIVALENCE (VECTX(2),CRITM) ,  (VECTY(2),RPM)
C
       COMMON     /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,XB,XR,YB,YT,
      +              DX,DY,IPLTF,IPRT2,IPRT3,RZ1(6),NZ(4),NCASE,NROT,
      +              NSTAT,IFLG,IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,NI,
      +              IPCNT,ISCNT
C
       COMMON     /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX,VECTY
C
       TYTL='ROTORDYNAMIC CRITICAL SPEED PLOT'
       N = NRMOD+NCMOD
C
       IF(IPLTF.EQ.0) THEN
C
C--------PLOT HEADER
C
         CALL HEADIN(.232)
C
C--------Y DIRECTION SPECS
C
         FNY=(XR-XB)/DX
         FACTY=DIMY/FNY
         VECTY(3)=XB
         VECTY(4)=DX
C
C--------X DIRECTION SPECS
C
         IF(DY.GE.1.) THEN
           FNX=(YT-YB)/DY
         ELSE
           FNX=ANINT(FNY*1.25)
C                                        NOTE: SCALE DESTROYS DX
           CALL SCALE(YB,FNX,2,1)
           YB=DX
           YT=YB+FNX*DY
         ENDIF
         VECTX(3)=YB
         VECTX(4)=DY
         FACTX=DIMX/FNX
C
C--------DRAW X/Y AXES
```

```
      C
              CALL AXES('NATURAL FREQUENCY - CYCLES/MINUTE',33,VECTX(3),
           +            VECTX(4),'ROTOR SPEN SPEED - RPM',22,VECTY(3),
           +            VECTY(4))
      C
      C-------FIX VECTX(4) FOR PLOTTING ON 180 DEGREE AXIS
      C       AND DISTORTED SCALE
      C
              VECTX(4) = -VECTX(4)*FACTY/FACTX
      C
      C-------DRAW SYNCHRONOUS LINE
      C
              VECTY(1) = AMAX1(XB,YB)
              IF(VECTY(1).GT.XR) GOTO 5
              VECTY(2) = AMIN1(YT,XR)
      C
              VECTX(1) = VECTY(1)
              VECTX(2) = VECTY(2)
      C
              CALL LINE(VECTX,VECTY,2,1,0,0)
      C
      C-------DRAW HALF-SYNCH. LINE
      C
          5   VECTY(1) = AMAX1(XB,YB*2)
              IF(VECTY(1).GT.XR) GOTO 10
              VECTY(2) = AMIN1(YT*2,XR)
      C
              VECTX(1) = VECTY(1)/2
              VECTX(2) = VECTY(2)/2
      C
              CALL LINE(VECTX,VECTY,2,1,0,0)
            ENDIF
      C
      C-----PLOT A SYMBOL AT EACH DATA POINT IN THE PLOT WINDOW
      C
         10 MHIGH=0
            DO 200 I=1,NI+1
1             RPM = RPM1 + DRPM*(I-1)
1             IF(RPM.GE.XB.AND.RPM.LE.XR) THEN
1               DO 100 J=1,N
2                 CRITM = W(2,J,I)*9.54929
2                 IF(CRITM.GE.YB.AND.CRITM.LE.YT) THEN
2                   IF(IPLTF.EQ.0) CALL LINE(CRITM,RPM,1,1,-1,11)
2                   MHIGH=MAX0(MHIGH,J)
2                 ENDIF
2       100     CONTINUE
1             ENDIF
1       200 CONTINUE
      C
            IF(IPLTF.EQ.0) THEN
      C
      C-------PLOT BORDER
      C
              CALL NEWPEN(2)
              CALL BOX(0.,0.,-FNX*FACTX/FACTY,FNY)
              CALL NEWPEN(1)
      C
      C-------PLOT Y DIRECTION DASHED LINES
      C
```

```
D Line# 1      7
                 CALL STDASH(.03760/FACTY,.03760/FACTY)
                 DO 25 I=1,INT(FNY)
1                  CALL PLOTD(0.,FLOAT(I),3)
1          25    CALL PLOTD(-DIMX/FACTY,FLOAT(I),2)
         C
         C-------PLOT X DIRECTION DASHED LINES
         C
                 DO 30 I=1,INT(FNX)
1                  CALL PLOTD(-I*FACTX/FACTY,0.,3)
1          30    CALL PLOTD(-I*FACTX/FACTY,FNY,2)
         C
         C-------END PLOT
         C
                 CALL PLOT(0.,0.,-999)
               ENDIF
         C
               END
```

Name	Type	Offset	P	Class
AMAX1				INTRINSIC
AMIN1				INTRINSIC
ANINT				INTRINSIC
CRITM	REAL	32		/DIMS /
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
DX	REAL	216		/DATA /
DY	REAL	220		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FLOAT				INTRINSIC
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
I	INTEGER*2	18		
IFLG	INTEGER*2	268		/DATA /
IFLG1	INTEGER*2	270		/DATA /
INT				INTRINSIC
IPCNT	INTEGER*2	294		/DATA /
IPLTF	INTEGER*2	224		/DATA /
IPRT	INTEGER*2	280		/DATA /
IPRT2	INTEGER*2	226		/DATA /
IPRT3	INTEGER*2	228		/DATA /
ISCNT	INTEGER*2	296		/DATA /
J	INTEGER*2	26		
KCRT	INTEGER*2	282		/DATA /
LMOD	INTEGER*2	4	*	
MAX0				INTRINSIC
MHIGH	INTEGER*2	8	*	
N	INTEGER*2	16		
NCASE	INTEGER*2	262		/DATA /
NCDOF	INTEGER*2	4		/DATA /
NCMOD	INTEGER*2	0		/DATA /
NI	INTEGER*2	292		/DATA /
NRDOF	INTEGER*2	6		/DATA /
NRMOD	INTEGER*2	2		/DATA /
NROT	INTEGER*2	264		/DATA /
NSTAT	INTEGER*2	266		/DATA /
NZ	INTEGER*2	254		/DATA /

```
D Line# 1      7
RPM     REAL            48   /DIMS   /
RPM1    REAL           284   /DATA   /
RZ1     REAL           230   /DATA   /
SC      REAL           276   /DATA   /
TH      REAL           272   /DATA   /
TITL    CHAR*1           8   /DATA   /
TYTL    CHAR*48        152   /DATA   /
VECTX   REAL            28   /DIMS   /
VECTY   REAL            44   /DIMS   /
W       REAL             0 *
XASP    REAL            24   /DIMS   /
XB      REAL           200   /DATA   /
XR      REAL           204   /DATA   /
YB      REAL           208   /DATA   /
YT      REAL           212   /DATA   /
```

```
*STBPLT.FOR*************************************************************
C
C      S U B R O U T I N E   S T B P L T
C
C      STBPLT EXECUTES THE STABILITY PLOT OPTION
C
C***********************************************************************
C
       SUBROUTINE STBPLT(W,LMOD,MHIGH)
C
       CHARACTER  TITL(72,2)*1,TYTL*48
C
       DIMENSION  W(2,LMOD,1),VECTX(4),VECTY(4),NZ1(18)
C
       EQUIVALENCE (REALM,VECTX(2)),(RPM,VECTY(2))
C
       EXTERNAL   XDISP,YDISP
C
       COMMON    /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,RZ(6),IPLTF,
      +               IPRT2,IPRT3,XB1,XR1,YB1,YT1,DX1,DY1,NZ(4),NCASE,
      +               NROT,NSTAT,IFLG,IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,
      +               NI,IPCNT,ISCNT
C
       COMMON    /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX,VECTY
C
C-----DEFINE THE PLOT SYMBOL CORRESPONDENCE WITH DISPLAY
C
       DATA NZ1/1,2,3,4,6,7,0,11,10,12,8,9,13,1,2,3,4,6/
C
C-----Y DIRECTION SPECS
C
       VECTY(3)=XB1
       VECTY(4)=DX1
       FNY=(XR1-XB1)/DX1
       FACTY=DIMY/FNY
C
C-----X DIRECTION SPECS
C
       VECTX(3)=YB1
       VECTX(4)=DY1
       FNX=(YT1-YB1)/DY1
       FACTX=DIMX/FNX
```

```
  C
  C-----FIX VECTX(4) FOR PLOTTING ON 180. DEGREE AXIS
  C
        VECTX(4) = -VECTX(4)
  C
  C-----M A I N   R O U T I N E
  C
        NPLTS = MHIGH/4. + 0.9
        IF(KCRT.EQ.1) NPLTS=1
  C
        DO 200 KOUNT = 1,NPLTS
1 C
1 C-------PLOT HEADER
1 C
1         TYTL='ROTORDYNAMIC STABILITY PLOT'
1         CALL HEADIN(.232)
1 C
1 C-------PLOT THE SYMBOL KEY
1 C
1         IF(KCRT.NE.1) THEN
1           DO 5 I=1,4
2             IF((KOUNT-1)*4+I.GT.MHIGH) GOTO 5
2             WRITE(TYTL,'(I2)') I+4*(KOUNT-1)
2             CALL SYMBOL(-5.65+(I-1)*.160,.271,.075,NZ1(NZ(I)),90.,-1)
2             CALL SYMBOL(-5.61+(I-1)*.160,.380,.118,'=',90.,1)
2             CALL SYMBOL(-5.61+(I-1)*.160,.540,.118,'MODE ',90.,5)
2             CALL SYMBOL(-5.61+(I-1)*.160,.990,.118,TYTL,90.,2)
2     5     CONTINUE
1           CALL SYMBOL(-5.76,.381,.118,'SYMBOLS',90.,7)
1           CALL BOX(-5.08,.200,-5.93,1.21)
1         ENDIF
1 C
1 C-------PLOT UNSTABLE/STABLE LABELS
1 C
1         CALL SYMBOL(-5.13,2.94,.119,'UNSTABLE',90.,8)
1         CALL BOX(-5.08,2.89,-5.28,3.69)
1         CALL SYMBOL(-.870,2.94,.119,'STABLE',90.,6)
1         CALL BOX(-.830,2.89,-1.02,3.51)
1 C
1 C-------DRAW X/Y AXES
1 C
1         CALL AXES('LAMBDA REAL - 1/SEC   (DAMPING FACTOR)',38,VECTX(3),
1      +            -VECTX(4),'ROTOR SPIN SPEED - RPM',22,VECTY(3),
1      +            VECTY(4))
1 C
1 C-------PLOT SYMBOLS AT DATA POINTS IN THE PLOT WINDOW
1 C
1         NZ9=11
1         N=MHIGH
1         DO 150 L=1,NI+1
2           RPM=RPM1+(L-1)*DRPM
2           IF(RPM.LT.XB1.OR.RPM.GT.XR1) GOTO 150
2           Y=YDISP(RPM)
2           IF(KCRT.NE.1) N=4
2           DO 100 I = 1,N
3             IK=(KOUNT-1)*4+I
3             IF(IK.GT.MHIGH) GO TO 150
3             IF(KCRT.NE.1) NZ9=NZ1(NZ(I))
3             REALM = W(1,IK,L)
```

```
D Line# 1      7                              Microsoft FORTRAN77 V3.13 8/05/83
3                IF(REALM.LT.YB1.OR.REALM.GT.YT1) GOTO 100
3                X=XDISP(REALM)
3     C-----------CHECK FOR PLOT KEY WINDOW
3                IF(KCRT.NE.1.AND.
3         +         (X.LE.-5.08.AND.Y.GE..200.AND.X.GE.-5.93.AND.Y.LE.1.21))
3         +                                                         GOTO 100
3     C-----------CHECK FOR UNSTABLE/STABLE LABEL WINDOWS
3              IF((X.LE.-5.08.AND.Y.GE.2.89.AND.X.GE.-5.28.AND.Y.LE.3.69).OR.
3         +       (X.LE.-.830.AND.Y.GE.2.89.AND.X.GE.-1.02.AND.Y.LE.3.51))
3         +                                                         GOTO 100
3                CALL SYMBOL(X/FACTY,Y/FACTY,.075/FACTY,NZ9,90.,-1)
3       100    CONTINUE
2       150  CONTINUE
1     C
1     C-------PLOT BORDER
1     C
1            CALL NEWPEN(2)
1            CALL BOX(0.,0.,-FNX*FACTX/FACTY,FNY)
1     C
1     C-------PLOT REALM=0 LINE
1     C
1            CALL PLOT(XDISP(0.)/FACTY,YDISP(XB1)/FACTY,3)
1            CALL PLOT(XDISP(0.)/FACTY,YDISP(XR1)/FACTY,2)
1            CALL NEWPEN(1)
1     C
1     C-------PLOT X DIRECTION DASHED LINES (AVOID PLOT LABELING)
1     C
1            CALL STDASH(.03760/FACTY,.03760/FACTY)
1            DO 10 I=1,INT(FNY)
2              Y=FLOAT(I)*FACTY
2              CALL PLOTD(0.,Y/FACTY,3)
2              IF(KCRT.NE.1.AND.(Y.GT..200.AND.Y.LT.1.21)) THEN
2                  CALL PLOTD(-5.08/FACTY,Y/FACTY,2)
2                  CALL PLOTD(-5.93/FACTY,Y/FACTY,3)
2              ENDIF
2              IF(Y.GT.2.89.AND.Y.LT.3.51) THEN
2                CALL PLOTD(-.830/FACTY,Y/FACTY,2)
2                CALL PLOTD(-1.02/FACTY,Y/FACTY,3)
2              ENDIF
2              IF(Y.GT.2.89.AND.Y.LT.3.69) THEN
2                CALL PLOTD(-5.08/FACTY,Y/FACTY,2)
2                CALL PLOTD(-5.28/FACTY,Y/FACTY,3)
2              ENDIF
2        10  CALL PLOTD(-DIMX/FACTY,Y/FACTY,2)
1     C
1     C-------PLOT Y DIRECTION DASHED LINES (AVOID PLOT LABELING)
1     C
1            DO 15 I=1,INT(FNX)
2              X=-FLOAT(I)*FACTX
2              CALL PLOTD(X/FACTY,0.,3)
2              IF(KCRT.NE.1.AND.(X.GT.-5.93.AND.X.LT.-5.08)) THEN
2                CALL PLOTD(X/FACTY,.200/FACTY,2)
2                CALL PLOTD(X/FACTY,1.21/FACTY,3)
2              ENDIF
2              IF(X.GT.-5.28.AND.X.LT.-5.08) THEN
2                CALL PLOTD(X/FACTY,2.89/FACTY,2)
2                CALL PLOTD(X/FACTY,3.69/FACTY,3)
2              ENDIF
2              IF(X.GT.-1.02.AND.X.LT.-.830) THEN
```

D Line# 1 7 Microsoft FORTRAN77 V3.13 8/05/83

```
2              CALL PLOTD(X/FACTY,2.89/FACTY,2)
2              CALL PLOTD(X/FACTY,3.51/FACTY,3)
2            ENDIF
2       15   CALL PLOTD(X/FACTY,FNY,2)
1     C
1      200 CALL PLOT(0.,0.,-999)
      C
           END
```

Name	Type	Offset	P	Class
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
DX1	REAL	246		/DATA /
DY1	REAL	250		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FLOAT				INTRINSIC
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
I	INTEGER*2	92		
IFLG	INTEGER*2	268		/DATA /
IFLG1	INTEGER*2	270		/DATA /
IK	INTEGER*2	120		
INT				INTRINSIC
IPCNT	INTEGER*2	294		/DATA /
IPLTF	INTEGER*2	224		/DATA /
IPRT	INTEGER*2	280		/DATA /
IPRT2	INTEGER*2	226		/DATA /
IPRT3	INTEGER*2	228		/DATA /
ISCNT	INTEGER*2	296		/DATA /
KCRT	INTEGER*2	282		/DATA /
KOUNT	INTEGER*2	84		
L	INTEGER*2	98		
LMOD	INTEGER*2	4	*	
MHIGH	INTEGER*2	8	*	
N	INTEGER*2	96		
NCASE	INTEGER*2	262		/DATA /
NCDOF	INTEGER*2	4		/DATA /
NCMOD	INTEGER*2	0		/DATA /
NI	INTEGER*2	292		/DATA /
NPLTS	INTEGER*2	82		
NRDOF	INTEGER*2	6		/DATA /
NRMOD	INTEGER*2	2		/DATA /
NROT	INTEGER*2	264		/DATA /
NSTAT	INTEGER*2	266		/DATA /
NZ	INTEGER*2	254		/DATA /
NZ1	INTEGER*2	46		
NZ9	INTEGER*2	94		
REALM	REAL	32		/DIMS /
RPM	REAL	48		/DIMS /
RPM1	REAL	284		/DATA /
RZ	REAL	200		/DATA /
SC	REAL	276		/DATA /
TH	REAL	272		/DATA /
TITL	CHAR*1	8		/DATA /
TYTL	CHAR*48	152		/DATA /
VECTX	REAL	28		/DIMS /

```
D Line# 1      7
VECTY   REAL          44   /DIMS   /
W       REAL           0 *
X       REAL         122
XASP    REAL          24   /DIMS   /
XB1     REAL         230   /DATA   /
XDISP                      EXTERNAL
XR1     REAL         234   /DATA   /
Y       REAL         106
YB1     REAL         238   /DATA   /
YDISP                      EXTERNAL
YT1     REAL         242   /DATA   /
```

```
*BOX.FOR*****************************************************************
C
C     S U B R O U T I N E   B O X
C
C************************************************************************
C
      SUBROUTINE BOX(X1,Y1,X2,Y2)
C
      CALL PLOT(X1,Y1,3)
      CALL PLOT(X2,Y1,2)
      CALL PLOT(X2,Y2,2)
      CALL PLOT(X1,Y2,2)
      CALL PLOT(X1,Y1,2)
C
      END
```

```
Name    Type          Offset P Class

X1      REAL              0 *
X2      REAL              8 *
Y1      REAL              4 *
Y2      REAL             12 *
```

```
*HEADIN*****************************************************************
C
C     S U B R O U T I N E   H E A D I N
C
C************************************************************************
C
      SUBROUTINE HEADIN(HIGHT)
C
      CHARACTER TITL(72,2),TYTL(48)
C
      COMMON    /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,RZ(6),IPLTF,
     +                IPRT2,IPRT3,XB1,XR1,YB1,YT1,DX1,DY1,NZ(4),NCASE,
     +                NROT,NSTAT,IFLG,IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,
     +                NI,IPCNT,ISCNT
C
      COMMON    /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),
     +                VECTY(4)
C
C-----DETERMINE LENGTH OF PLOT TITLE FOR CENTERING
C
      L=48
    5 IF(L.NE.0.AND.(TYTL(L).EQ.' '.OR.TYTL(L).EQ.'$')) THEN
```

```
             L=L-1
             GOTO 5
          ENDIF
      C
      C-----PLOT HEADER
      C
          CALL ASPECT(XASP)
          CALL SYMBOL(2.03,(8-L*HIGHT*.872*XASP)/2,HIGHT,TYTL,90.,L)
          CALL PLOT(2.07,(8-L*HIGHT*.872*XASP)/2,3)
          CALL PLOT(2.07,(8+L*HIGHT*.872*XASP)/2,2)
          CALL SYMBOL(2.30,1.58,.120,TITL,90.,60)
          CALL SYMBOL(2.52,1.58,.120,TITL(1,2),90.,60)
      C
      C-----SET PLOT ORIGIN
      C
          CALL PLOT(8.90,1.58,-3)
      C
          END
```

Name	Type	Offset	P	Class
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
DX1	REAL	246		/DATA /
DY1	REAL	250		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
HIGHT	REAL	0	*	
IFLG	INTEGER*2	268		/DATA /
IFLG1	INTEGER*2	270		/DATA /
IPCNT	INTEGER*2	294		/DATA /
IPLTF	INTEGER*2	224		/DATA /
IPRT	INTEGER*2	280		/DATA /
IPRT2	INTEGER*2	226		/DATA /
IPRT3	INTEGER*2	228		/DATA /
ISCNT	INTEGER*2	296		/DATA /
KCRT	INTEGER*2	282		/DATA /
L	INTEGER*2	158		
NCASE	INTEGER*2	262		/DATA /
NCDOF	INTEGER*2	4		/DATA /
NCMOD	INTEGER*2	0		/DATA /
NI	INTEGER*2	292		/DATA /
NRDOF	INTEGER*2	6		/DATA /
NRMOD	INTEGER*2	2		/DATA /
NROT	INTEGER*2	264		/DATA /
NSTAT	INTEGER*2	266		/DATA /
NZ	INTEGER*2	254		/DATA /
RPM1	REAL	284		/DATA /
RZ	REAL	200		/DATA /
SC	REAL	276		/DATA /
TH	REAL	272		/DATA /
TITL	CHAR*1	8		/DATA /
TYTL	CHAR*1	152		/DATA /
VECTX	REAL	28		/DIMS /
VECTY	REAL	44		/DIMS /
XASP	REAL	24		/DIMS /

```
XB1     REAL              230   /DATA   /
XR1     REAL              234   /DATA   /
YB1     REAL              238   /DATA   /
YT1     REAL              242   /DATA   /
```

```
*AXES*******************************************************************
C
C     S U B R O U T I N E   A X E S
C
C***********************************************************************
C
      SUBROUTINE AXES(XTIT,LX,XB,XD,YTIT,LY,YB,YD)
C
      CHARACTER XTIT(LX),YTIT(LY)
C
      COMMON    /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),
     +                VECTY(4)
C
C-----DRAW X/Y AXES
C
      CALL FACTOR(FACTX)
      CALL STAXIS(.09/FACTX,.124/FACTX,.075/FACTX,.07/FACTX,0)
      CALL AXIS(0.,0.,XTIT,LX,FNX,180.,XB,XD)
C
      CALL FACTOR(FACTY)
      CALL STAXIS(.09/FACTY,.124/FACTY,.075/FACTY,.07/FACTY,0)
      CALL AXIS(0.,0.,YTIT,-LY,FNY,90.,YB,YD)
C
      END
```

```
Name    Type          Offset P Class

DIMX    REAL              0     /DIMS   /
DIMY    REAL              4     /DIMS   /
FACTX   REAL             16     /DIMS   /
FACTY   REAL             20     /DIMS   /
FNX     REAL              8     /DIMS   /
FNY     REAL             12     /DIMS   /
LX      INTEGER*2         4 *
LY      INTEGER*2        20 *
VECTX   REAL             28     /DIMS   /
VECTY   REAL             44     /DIMS   /
XASP    REAL             24     /DIMS   /
XB      REAL              8 *
XD      REAL             12 *
XTIT    CHAR*1            0 *
YB      REAL             24 *
YD      REAL             28 *
YTIT    CHAR*1           16 *
```

```
*XDISP******************************************************************
C
C     F U N C T I O N   X D I S P
C
C***********************************************************************
C
      FUNCTION XDISP(X)
```

```
C
      COMMON /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),VECTY(4)
C
      XDISP=(X-VECTX(3))/VECTX(4)*FACTX
C
      END
```

Name	Type	Offset	P	Class
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
VECTX	REAL	28		/DIMS /
VECTY	REAL	44		/DIMS /
X	REAL	0	*	
XASP	REAL	24		/DIMS /

```
*YDISP*********************************************************************
C
C     F U N C T I O N   Y D I S P
C
C**************************************************************************
C
      FUNCTION YDISP(Y)
C
      COMMON /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),VECTY(4)
C
      YDISP=(Y-VECTY(3))/VECTY(4)*FACTY
C
      END
```

Name	Type	Offset	P	Class
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
VECTX	REAL	28		/DIMS /
VECTY	REAL	44		/DIMS /
XASP	REAL	24		/DIMS /
Y	REAL	0	*	

Name	Type	Size	Class
ASPECT			SUBROUTINE
AXES			SUBROUTINE
AXIS			SUBROUTINE
BOX			SUBROUTINE
CRTPLT			SUBROUTINE
DATA		298	COMMON
DIMS		60	COMMON

```
FACTOR                      SUBROUTINE
FNCPLT                      SUBROUTINE
HEADIN                      SUBROUTINE
LINE                        SUBROUTINE
NEWPEN                      SUBROUTINE
PLOT                        SUBROUTINE
PLOTD                       SUBROUTINE
SCALE                       SUBROUTINE
STAXIS                      SUBROUTINE
STBPLT                      SUBROUTINE
STDASH                      SUBROUTINE
SYMBOL                      SUBROUTINE
XDISP   REAL                FUNCTION
YDISP   REAL                FUNCTION

 Pass One     No Errors Detected
             480 Source Lines
```

```
*SHPPLT.FOR**********************************************************
C
C     S U B R O U T I N E   S H P P L T
C
C     ROTORDYNAMICS INTERACTIVE MODE SHAPE PLOTTING
C
C********************************************************************
C
C          RYR, RYI, RZR, RZI    =  ROTOT Y AND Z  REAL AND IMAG
C          CYR, CYI, CZR, CZI    =  CASING Y AND Z REAL AND IMAG
C          PHIR,PHIC =  WHIRL DIRECTION  FWD=FORWARD, REV=BACKWARD
C          NSTAT  =  NO. OF STATIONS, .LE. 10
C          IFLG   =  CONTROL FLAG....  =1  PLOTS ROTOR ONLY,
C                                      =2  PLOTS CASING ONLY,
C                                      =3  PLOTS ROTOR, CASING, AND
C                                          ROTOR-CASING
C          IFLG1  =  UNUSED
C
C          TH  =  VIEWING ANGLE (DEGREES)
C          SC  =  SCALE FACTOR
C
C********************************************************************
C
$STORAGE:2
C
      SUBROUTINE SHPPLT(VECT,Z,AC,AR,ICASE,IROT,LMOD,LCDOF,LRDOF,X)
C
      CHARACTER  TITL(72,2),PREC(2)*3,TYTL*48
C
      LOGICAL    PHIR(10),PHIC(10)
C
      EXTERNAL   CABS,XDISP,YDISP
C
      EQUIVALENCE (IPHIC,PHIC),(IPHIR,PHIR)
C
      DIMENSION  AC(LCDOF,1),AR(LRDOF,1),ICASE(1),IROT(1),VECT(2,1),
     +           Z(2,1),X(1)
C
      DIMENSION  CYBAR(250),CZBAR(250),RYR(10),RYI(10),RZR(10),RZI(10),
     1           RYBAR(250),RZBAR(250),CYR(10),CYI(10),CZR(10),CZI(10),
     2           YCL(10),YREL(250),IPHIC(10),IPHIR(10),ZCL(10),WZ(2),
     3           ZREL(250)
C
      COMMON    /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,RZ(6),IPLTF,
     +                IPRT2,IPRT3,RZ1(6),NZ(4),NCASE,NROT,NSTAT,IFLG,
     +                IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,NI,IPCNT,ISCNT
C
      COMMON     /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),
     +                 VECTY(4)
C
      DATA       PREC/'REV','FWD'/ , I1,I2/1,2/
C
      NMODES=NRMOD+NCMOD
C
C-----DEFINE DATA FOR MODE SHAPE PLOTTING
C
      IF(IPRT3 .EQ. 0) THEN
        THETA=TH*.01745
        DO 5 I=I1,NSTAT
```

```
           YCL(I) = X(I) * COS(THETA)
           ZCL(I) = X(I) * SIN(THETA)
    5    CONTINUE
         CALL STDASH(.0376,.0376)
         XL = X(NSTAT) - X(I1)
         FACTX=8.2
         FACTY=4.6
         DIMX = FACTX-I2*SC*XL
         DIMY = FACTY-I2*SC*XL
         IF(ABS(SIN(THETA)).LT.DIMY/CABS(DIMX)) THEN
           VECTX(3)=YCL(NSTAT)-SC*XL
           VECTX(4)=YCL(I1)+SC*XL-VECTX(3)
           VECTY(4)=VECTX(4)*FACTY/FACTX
           VECTY(3)=(ZCL(NSTAT)+ZCL(I1)-VECTY(4))/I2
         ELSE
           VECTY(3)=ZCL(NSTAT)-SC*XL
           VECTY(4)=ZCL(I1)+SC*XL-VECTY(3)
           VECTX(4)=VECTY(4)*FACTX/FACTY
           VECTX(3)=(YCL(NSTAT)+YCL(I1)-VECTX(4))/I2
         ENDIF
       ENDIF
C
C-----M A I N   R O U T I N E
C
   10 READ(3) RPM,JMODE,WZ,((Z(J,I),J=I1,I2),I=I1,NMODES)
       IF(JMODE.LT.0) RETURN
       WRITE(6,'(1H1)')
       IF(RPM.NE.RPMLST) THEN
         WRITE(6,8001)
         WRITE(6,8000) TITL,RPM
         RPMLST=RPM
       ENDIF
C
C------BACKTRANSFORM THE ROTOR DOFS
C
       DO 15 I = I1,NROT
        VECT(I1,I)=0.0
        VECT(I2,I)=0.0
        DO 15 J=I1,NRMOD
         VECT(I1,I) = VECT(I1,I)  +  AR(IROT(I),J)*Z(I1,J)
   15    VECT(I2,I) = VECT(I2,I)  +  AR(IROT(I),J)*Z(I2,J)
C
C------BACKTRANSFORM THE CASING DOFS
C
       DO 20 I = I1,NCASE
        VECT(I1,NROT+I)=0.0
        VECT(I2,NROT+I)=0.0
        DO 20 J  = I1,NCMOD
         VECT(I1,NROT+I) = VECT(I1,NROT+I) + AC(ICASE(I),J)*Z(I1,NRMOD+J)
   20    VECT(I2,NROT+I) = VECT(I2,NROT+I) + AC(ICASE(I),J)*Z(I2,NRMOD+J)
C
       XE = WZ(I2) * 9.54929
C
C------WRITE MODE SHAPE
C
       IF(IPRT.EQ.0) THEN
C......NORMALIZE VECTOR WITH RESPECT TO ALL DOF'S
        PMAX  = 0.0
        DO 25 I  =  I1,NROT+NCASE
```

```
1      25  PMAX=AMAX1(PMAX,CABS(VECT(I1,I)))
         C
             WRITE(6,2000) JMODE,WZ,XE
         C
             IF(NROT .NE. 0) THEN
              WRITE(6,2500)
              WRITE(6,4010)
              WRITE(6,2150)  (IROT(I),CABS(VECT(I1,I))/PMAX,ATAN2(VECT(I2,I),
         +                   VECT(I1,I)),(VECT(J,I)/PMAX,J=I1,I2),I=I1,NROT)
             ENDIF
         C
             IF(NCASE .NE. 0) THEN
              WRITE(6,2600)
              WRITE(6,4010)
              WRITE(6,2150)  (ICASE(I),CABS(VECT(I1,NROT+I))/PMAX,ATAN2(VECT(I2
         +                   ,NROT+I),VECT(I1,NROT+I)),(VECT(K,NROT+I)/PMAX,
         +                   K=I1,I2),I=I1,NCASE)
             ENDIF
         C
            ENDIF
         C
         C-----PLOT MODE SHAPE
         C
            IF(IPRT3.NE.0) GOTO 10
         C
         C.....NORMALIZE WITH RESPECT TO STATION DOF'S ONLY
            PMAX=0.0
            DO 30 I=I1,NSTAT
1           PMAX=AMAX1(PMAX,VECT(I1,I*I2-I1),VECT(I1,I*I2),
1          +          VECT(I1,NROT+I*I2-I1),VECT(I1,NROT+I*I2))
1      30  PMAX=AMAX1(PMAX,VECT(I2,I*I2-I1),VECT(I2,I*I2),
1          +          VECT(I2,NROT+I*I2-I1),VECT(I2,NROT+I*I2))
         C
            DO 35 I=I1,NSTAT
1            RYR(I) = VECT(I1,I*I2-I1)/PMAX        * XL * SC
1            RYI(I) = VECT(I2,I*I2-I1)/PMAX        * XL * SC
1            RZR(I) = VECT(I1,I*I2)/PMAX           * XL * SC
1            RZI(I) = VECT(I2,I*I2)/PMAX           * XL * SC
1            CYR(I) = VECT(I1,NROT+I*I2-I1)/PMAX   * XL * SC
1            CYI(I) = VECT(I2,NROT+I*I2-I1)/PMAX   * XL * SC
1            CZR(I) = VECT(I1,NROT+I*I2)/PMAX      * XL * SC
1            CZI(I) = VECT(I2,NROT+I*I2)/PMAX      * XL * SC
1      35  CONTINUE
         C
            WRITE(6,95) RPM,JMODE,XE
         C
         C-----ROTOR GROUP MODAL DATA
         C
            IF(IFLG.NE.I2) THEN
             WRITE(6,1160)
             WRITE(6,1159)
             DO 40      I     =    I1,NSTAT
1             PHIR(I) = RYI(I)*RZR(I) - RYR(I)*RZI(I)   .GE.   0.0D0
1      40    WRITE(6,1161) I,RYR(I),RYI(I),RZR(I),RZI(I),PREC(IPHIR(I)+I1)
            ENDIF
         C
         C-----CASING GROUP MODAL DATA
         C
            IF(IFLG .NE. I1) THEN
```

```
       WRITE(6,1190)
       WRITE(6,1159)
       DO 50    I     =    I1,NSTAT
         PHIC(I) = CYI(I)*CZR(I) - CYR(I)*CZI(I)   .GE.   0.0D0
   50  WRITE(6,1161) I,CYR(I),CYI(I),CZR(I),CZI(I),PREC(IPHIC(I)+I1)
       ENDIF
C
C-----COMPUTE ORBIT DATA FOR PLOTTING
C
       DO 60     I  =  I1,NSTAT
         DO 60      J  =  I1,25
           N  =  J + 25*(I-I1)
           PRECS  =  .2618*(J-I1)
           IF(IFLG.NE.I2) THEN
             Y1 =  RYR(I)*COS(PRECS) - RYI(I)*SIN(PRECS)
             Z1 =  RZR(I)*COS(PRECS) - RZI(I)*SIN(PRECS)
             YREL(N)   =  Y1
             ZREL(N)   =  Z1
             RYBAR(N)   =   YCL(I)   +   Y1
             RZBAR(N)   =   ZCL(I)   +   Z1
           ENDIF
           IF(IFLG.NE.I1) THEN
             Y1 =  CYR(I)*COS(PRECS) - CYI(I)*SIN(PRECS)
             Z1 =  CZR(I)*COS(PRECS) - CZI(I)*SIN(PRECS)
             CYBAR(N)   =   YCL(I)   +   Y1
             CZBAR(N)   =   ZCL(I)   +   Z1
           ENDIF
           IF(IFLG.EQ.3) THEN
             YREL(N)  =  (YREL(N) - Y1) + YCL(I)
             ZREL(N)   =  (ZREL(N) - Z1)  + ZCL(I)
           ENDIF
   60  CONTINUE
C
C-----PLOT MODE SHAPES
C
       IF(IFLG.NE.I2) THEN
         TYTL='Rotor Group Mode Shape'
         CALL SPLOT(PHIR(I1),YCL,ZCL,RYBAR,RZBAR,RPM,XE)
         CALL PLOT(0.,0.,-999)
       ENDIF
C
       IF(IFLG .NE. I1) THEN
         TYTL='Casing Group Mode Shape'
         CALL SPLOT(PHIC(I1),YCL,ZCL,CYBAR,CZBAR,RPM,XE)
         CALL PLOT(0.,0.,-999)
       ENDIF
C
       IF(IFLG.EQ.3) THEN
         TYTL='Relative Deflection Mode Shape'
         CALL SPLOT(I2,YCL,ZCL,YREL,ZREL,RPM,XE)
         CALL PLOT(0.,0.,-999)
       ENDIF
C
       GOTO 10
C
C-----FORMAT STATEMENTS
C
   95  FORMAT(///,5X,'RPM=',F12.3,5X,'MODE=',I3,5X,'CPM=',F12.3)
 1159  FORMAT(5X,'STATION',5X,'YR',12X,'YI',12X,'ZR',12X,'ZI',6X,
```

```
     +          'PRECESSION'/)
1160 FORMAT(///,10X,'*** ROTOR GROUP MODE SHAPE PLOTTING DATA ***',//)
1161 FORMAT(7X,I2,1X,1P4E14.5,4X,A)
1190 FORMAT(///,10X,'*** CASING GROUP MODE SHAPE PLOTTING DATA ***',//)
2000 FORMAT(///5X,'MODE',I3,4X,'REAL=',1PE12.5,4X,'IMAG=',
    1          E12.5,4X,'CPM =',0PF10.3)
2150 FORMAT(10X,I3,1X,1P4E15.5)
2500 FORMAT(///10X,'*** ROTOR GROUP MODAL DISPLACEMENT DATA ***'//)
2600 FORMAT(///10X,'*** CASING GROUP MODAL DISPLACEMENT DATA ***'//)
4010 FORMAT(10X,'DOF',6X,'AMPLITUDE',5X,'PHASE ANG.',8X,'REAL',9X,
    1          'IMAGINARY'/)
8000 FORMAT(/5X,72A//5X,72A///5X,'ROTOR SPIN SPEED =',F8.0,'  RPM')
8001 FORMAT(//5X,'*** ROTORDYNAMICS MODE SHAPE PLOTTING PROGRAM ***'//)
C
      END
```

Name	Type	Offset	P	Class
ABS				INTRINSIC
AC	REAL	8	*	
AMAX1				INTRINSIC
AR	REAL	12	*	
ATAN2				INTRINSIC
CABS				EXTERNAL
COS				INTRINSIC
CYBAR	REAL	4296		
CYI	REAL	6416		
CYR	REAL	6376		
CZBAR	REAL	5296		
CZI	REAL	6336		
CZR	REAL	6296		
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
I	INTEGER*2	6466		
I1	INTEGER*2	6456		
I2	INTEGER*2	6458		
ICASE	INTEGER*2	16	*	
IFLG	INTEGER*2	268		/DATA /
IFLG1	INTEGER*2	270		/DATA /
IPCNT	INTEGER*2	294		/DATA /
IPHIC	INTEGER*2	4276		
IPHIR	INTEGER*2	4256		
IPLTF	INTEGER*2	224		/DATA /
IPRT	INTEGER*2	280		/DATA /
IPRT2	INTEGER*2	226		/DATA /
IPRT3	INTEGER*2	228		/DATA /
IROT	INTEGER*2	20	*	
ISCNT	INTEGER*2	296		/DATA /
J	INTEGER*2	6488		
JMODE	INTEGER*2	6486		
K	INTEGER*2	6548		
KCRT	INTEGER*2	282		/DATA /
LCDOF	INTEGER*2	28	*	
LMOD	INTEGER*2	24	*	

```
D Line# 1     7
LRDOF   INTEGER*2      32 *
N       INTEGER*2    6580
NCASE   INTEGER*2     262   /DATA   /
NCDOF   INTEGER*2       4   /DATA   /
NCMOD   INTEGER*2       0   /DATA   /
NI      INTEGER*2     292   /DATA   /
NMODES  INTEGER*2    6460
NRDOF   INTEGER*2       6   /DATA   /
NRMOD   INTEGER*2       2   /DATA   /
NROT    INTEGER*2     264   /DATA   /
NSTAT   INTEGER*2     266   /DATA   /
NZ      INTEGER*2     254   /DATA   /
PHIC    LOGICAL*2    4276
PHIR    LOGICAL*2    4256
PMAX    REAL         6526
PREC    CHAR*3       2090
PRECS   REAL         6582
RPM     REAL         6482
RPM1    REAL          284   /DATA   /
RPMLST  REAL         6494
RYBAR   REAL         3256
RYI     REAL         3216
RYR     REAL         2096
RZ      REAL          200   /DATA   /
RZ1     REAL          230   /DATA   /
RZBAR   REAL         2216
RZI     REAL         2176
RZR     REAL         2136
SC      REAL          276   /DATA   /
SIN                         INTRINSIC
TH      REAL          272   /DATA   /
THETA   REAL         6462
TITL    CHAR*1          8   /DATA   /
TYTL    CHAR*48       152   /DATA   /
VECT    REAL            0 *
VECTX   REAL           28   /DIMS   /
VECTY   REAL           44   /DIMS   /
WZ      REAL         2082
X       REAL           36 *
XASP    REAL           24   /DIMS   /
XDISP                       EXTERNAL
XE      REAL         6522
XL      REAL         6474
Y1      REAL         6586
YCL     REAL         2042
YDISP                       EXTERNAL
YREL    REAL         1042
Z       REAL            4 *
Z1      REAL         6590
ZCL     REAL            2
ZREL    REAL           42

   252 *SPLOT*********************************************************************
   253 C
   254 C      S U B R O U T I N E   S P L O T
   255 C
   256 C**********************************************************************
   257 C
```

```
      SUBROUTINE SPLOT(NPHI,YCL,ZCL,RYBAR,RZBAR,RPM,XE)
C
      DIMENSION YCL(1),ZCL(1),RYBAR(1),RZBAR(1),NPHI(1)
C
      CHARACTER TITL(72,2),TYTL(48),MESSAG(5)*31,M(31,2),M1*10,M2*10
C
      EQUIVALENCE (MESSAG,M),(M(17,1),M1),(M(17,2),M2)
C
      EXTERNAL    XDISP,YDISP
C
      COMMON     /DATA/NCMOD,NRMOD,NCDOF,NRDOF,TITL,TYTL,RZ(6),IPLTF,
     +                IPRT2,IPRT3,RZ1(6),NZ(4),NCASE,NROT,NSTAT,IFLG,
     +                IFLG1,TH,SC,IPRT,KCRT,RPM1,DRPM,NI,IPCNT,ISCNT
C
      COMMON     /DIMS/DIMX,DIMY,FNX,FNY,FACTX,FACTY,XASP,VECTX(4),
     +                VECTY(4)
C
      DATA MESSAG/'Spin Speed                  RPM' ,
     +            'Nat Frequency               CPM' ,
     +            ' ORBIT - BACKWARD PRECESSION   ' ,
     +            ' ORBIT - FORWARD PRECESSION    ' ,
     +            '                               ' /
      DATA HIGHT/.220/
C
C-----DETERMINE LENGTH OF PLOT TITLE FOR CENTERING
C
      L=48
    5 IF(L.NE.O.AND.TYTL(L).EQ.' ') THEN
        L=L-1
        GOTO 5
      ENDIF
C
C-----PLOT HEADER
C
      CALL ASPECT(XASP)
C
      RATIO=.872*XASP
      CALL SYMBOL((11-L*HIGHT*RATIO)/2,6.74,HIGHT,TYTL,0.,L)
      CALL PLOT((11-L*HIGHT*RATIO)/2,6.70,3)
      CALL PLOT((11+L*HIGHT*RATIO)/2,6.70,2)
      CALL SYMBOL((11-L*HIGHT*RATIO)/2,6.52,.13,TITL,0.,60)
      CALL SYMBOL((11-L*HIGHT*RATIO)/2,6.34,.13,TITL(1,2),0.,60)
C
      WRITE(M1,9000) RPM
      CALL SYMBOL(1.93,5.48,.119,MESSAG(1),0.,31)
      WRITE(M2,9000) XE
      CALL SYMBOL(1.93,5.24,.119,MESSAG(2),0.,31)
C
C-----DRAW AXES
C
      CALL PLOT(9.,2.2,3)
      CALL SYMBOL(9.3,2.2,.1,2,270.,-2)
      CALL SYMBOL(9.36,2.18,.150,'y',0.,1)
C
      CALL PLOT(9.,2.2,3)
      CALL SYMBOL(9.,2.5,.1,2,0.,-2)
      CALL SYMBOL(8.98,2.57,.150,'z',0.,1)
C
      THETA=TH*.01745
```

```
D Line# 1       7                                  Microsoft FORTRAN77 V3.13 8/05/83
                CALL PLOT(9.,2.2,3)
                CALL SYMBOL(9.+.3*COS(THETA),2.2+.3*SIN(THETA),.1,2,TH-90.,-2)
                CALL SYMBOL(9.+.44*COS(THETA),2.2+.44*SIN(THETA),.150,'x',0.,1)
          C
          C-----SET PLOT ORIGIN
          C
                CALL PLOT(1.4,1.,-3)
          C
          C-----WRITE STATION NUMBERS
          C
                DO 70 I = 1,NSTAT
1                 WRITE(M1,9001) I
1                 CALL SYMBOL(XDISP(YCL(I))-.04,YDISP(ZCL(I))-.16,.12,M1,0.,2)
1            70 CONTINUE
          C
          C-----PLOT THE MODE SHAPE
          C
                CALL PLOT(XDISP(RYBAR),YDISP(RZBAR),3)
                DO 75 I=2,25*NSTAT
1                 CALL PLOT(XDISP(RYBAR(I)),YDISP(RZBAR(I)),2)
1            75 CONTINUE
          C
          C-----PLOT THE PRECESSION ARROWS
          C
                IF(NPHI(1).NE.2) THEN
                  K=0
                  CALL NEWPEN(2)
                  DO 80 I=1,NSTAT
1                   J=(I-1)*25+13
1                   X=RYBAR(J)
1                   XD=RYBAR(J+1)-X
1                   Y=RZBAR(J)
1                   YD=RZBAR(J+1)-Y
1                   IF(XD.NE.0..OR.YD.NE.0.) THEN
1                     THETA=AMOD(ATAN2(-XD,YD)/.01745+360.,360.)
1                     CALL SYMBOL(XDISP(X),YDISP(Y),.090,2,THETA,-1)
1                     IF(K.EQ.0) K=I
1                   ENDIF
1            80   CONTINUE
                  CALL NEWPEN(1)
                  IF(K.NE.0) THEN
                    WRITE(M1,'(8HSTATION ,I2)') K
                    CALL SYMBOL(3.73,.35,.119,M1,0.,10)
                    CALL SYMBOL(999.,999.,.119,MESSAG(NPHI(K)+3),0.,31)
                  ENDIF
                ENDIF
          C
                CALL PLOTD(XDISP(YCL),YDISP(ZCL),3)
                DO 85 I=2,NSTAT
1                 CALL PLOTD(XDISP(YCL(I)),YDISP(ZCL(I)),2)
1            85 CONTINUE
          C
          C-----FORMAT
          C
           9000 FORMAT(F10.3)
           9001 FORMAT(I2)
                END
```

Name	Type	Offset	P	Class
AMOD				INTRINSIC
ATAN2				INTRINSIC
COS				INTRINSIC
DIMX	REAL	0		/DIMS /
DIMY	REAL	4		/DIMS /
DRPM	REAL	288		/DATA /
FACTX	REAL	16		/DIMS /
FACTY	REAL	20		/DIMS /
FNX	REAL	8		/DIMS /
FNY	REAL	12		/DIMS /
HIGHT	REAL	7450		
I	INTEGER*2	7464		
IFLG	INTEGER*2	268		/DATA /
IFLG1	INTEGER*2	270		/DATA /
IPCNT	INTEGER*2	294		/DATA /
IPLTF	INTEGER*2	224		/DATA /
IPRT	INTEGER*2	280		/DATA /
IPRT2	INTEGER*2	226		/DATA /
IPRT3	INTEGER*2	228		/DATA /
ISCNT	INTEGER*2	296		/DATA /
J	INTEGER*2	7510		
K	INTEGER*2	7502		
KCRT	INTEGER*2	282		/DATA /
L	INTEGER*2	7454		
M	CHAR*1	7294		
M1	CHAR*10	7310		
M2	CHAR*10	7341		
MESSAG	CHAR*31	7294		
NCASE	INTEGER*2	262		/DATA /
NCDOF	INTEGER*2	4		/DATA /
NCMOD	INTEGER*2	0		/DATA /
NI	INTEGER*2	292		/DATA /
NPHI	INTEGER*2	0	*	
NRDOF	INTEGER*2	6		/DATA /
NRMOD	INTEGER*2	2		/DATA /
NROT	INTEGER*2	264		/DATA /
NSTAT	INTEGER*2	266		/DATA /
NZ	INTEGER*2	254		/DATA /
RATIO	REAL	7456		
RPM	REAL	20	*	
RPM1	REAL	284		/DATA /
RYBAR	REAL	12	*	
RZ	REAL	200		/DATA /
RZ1	REAL	230		/DATA /
RZBAR	REAL	16	*	
SC	REAL	276		/DATA /
SIN				INTRINSIC
TH	REAL	272		/DATA /
THETA	REAL	7460		
TITL	CHAR*1	8		/DATA /
TYTL	CHAR*1	152		/DATA /
VECTX	REAL	28		/DIMS /
VECTY	REAL	44		/DIMS /
X	REAL	7512		
XASP	REAL	24		/DIMS /
XD	REAL	7516		

```
XDISP                          EXTERNAL
XE      REAL           24 *
Y       REAL         7520
YCL     REAL            4 *
YD      REAL         7524
YDISP                          EXTERNAL
ZCL     REAL            8 *
```

```
*CABS.FOR**************************************************************
C
C     R E A L   F U N C T I O N   C A B S
C
C**********************************************************************
C
      REAL FUNCTION CABS(Z)
C
      REAL*4 Z(1)
C
      CABS=DSQRT(DBLE(Z(1)*Z(1)+Z(2)*Z(2)))
C
      END
```

```
Name    Type          Offset P Class

DBLE                           INTRINSIC
DSQRT                          INTRINSIC
Z       REAL             0 *
```

```
Name    Type            Size   Class

ASPECT                         SUBROUTINE
CABS    REAL                   FUNCTION
DATA                     298   COMMON
DIMS                      60   COMMON
NEWPEN                         SUBROUTINE
PLOT                           SUBROUTINE
PLOTD                          SUBROUTINE
SHPPLT                         SUBROUTINE
SPLOT                          SUBROUTINE
STDASH                         SUBROUTINE
SYMBOL                         SUBROUTINE
XDISP   REAL                   FUNCTION
YDISP   REAL                   FUNCTION
```

```
 Pass One     No Errors Detected
            386 Source Lines
```

```
*LOGINIT***********************************************************************
C
C     P R O G R A M   L O G I N I T
C
C     ROTOR  DYNAMICS  ANALYSIS  PROGRAM
C
C******************************************************************************
C
$STORAGE:2
C
      PROGRAM LOGINIT
C
      CHARACTER   BANNR(3)*31,C1,INPUT*40,CULINE(80)
C
      INTEGER     NDATE(3),NTIME(4),IULINE(20)
C
      EQUIVALENCE (C1,INPUT),(CULINE(41),IULINE)
C
      DATA BANNR /'P R O G R A M   L O G I N I T',
     +            'ROTORDYNAMICS ANALYSIS',
     +            'JUNE 1984'/
      DATA CULINE /40*'_',40*' '/ , IULINE /20*#0808/
C
C-----SET UP SCREEN
C
      CALL QCLEAR(1,7)
      CALL QBORD(1)
      CALL QCSIZ(0,0)
      CALL QCMOV(0,24)
C
C-----PROGRAM BANNER
C
      CALL QDATE(NDATE(3),NDATE,NDATE(2))
      CALL QTIME(NTIME,NTIME(2),NTIME(3),NTIME(4))
      NTIME(3) = NTIME(3) + NTIME(4)/100. + 0.5
C
      WRITE(*,8004) BANNR
      WRITE(*,8005) (NDATE(I),I=1,3),(NTIME(I),I=1,3)
C
C-----VERIFY THAT LOG FILE IS TO BE UPDATED (DESTROYED)
C
      CALL QCSIZ(0,7)
      WRITE(*,9000) '  ***  THIS PROGRAM INITIALIZED (ERASES AND ',
     +              'RECREATES) THE RSTAB USAGE LOG  ***'
      WRITE(*,8001) 'DO YOU WANT TO ERASE AND RECREATE THE RSTAB ',
     +              'USAGE LOG [Y/N]? '
      READ(*,9000) INPUT
      IF(INPUT.EQ.' ') GOTO 1000
    5 IF(C1.EQ.' ') THEN
        READ(INPUT,'(1X,A)') INPUT
        GOTO 5
      ENDIF
      IF(C1.NE.'Y') GOTO 1000
C
C-----GET USER'S NAME
C
      WRITE(*,'(//2X,81A\)') 'ENTER YOUR NAME: ',CULINE
      READ(*,9000) INPUT
      IF(INPUT.EQ.' ') GOTO 1000
```

```
C
C-----CREATE LOG FILE
C
      OPEN(7,FILE='LOG.SAV',STATUS='NEW',ACCESS='DIRECT',
     +        FORM='UNFORMATTED',RECL=242)
      WRITE(7,REC=1) 1,NDATE,NTIME,INPUT
      CLOSE(7)
C
C-----NORMAL TERMINATION
C
      WRITE(*,9001) 'UPDATE COMPLETE'
      GOTO 2000
C
C-----NO UPDATE
C
 1000 WRITE(*,9001) 'NO UPDATE MADE'
C
C-----TERMINATION
C
 2000 CALL QCSIZ(0,0)
C
C-----FORMAT STATEMENTS
C
 8001 FORMAT(//2X,2A,1X\)
 8004 FORMAT(1H1,5(/),2X,78(1H*),3(/7H  *****,68X,5H*****/7H  *****,18X,
     +        A,19X,5H*****)/7H  *****,68X,5H*****/2X,78(1H*))
 8005 FORMAT(2X,5HDATE:,I3,1H/,I2,1H/,I4,42X,11HBEGIN TIME:,I3,
     +        2(1H:,I2),6(/))
 9000 FORMAT(72A)
 9001 FORMAT(//2X,A/)
C
      END
```

Name	Type	Offset	P	Class
BANNR	CHAR*31	2		
C1	CHAR*1	95		
CULINE	CHAR*1	150		
I	INTEGER*2	230		
INPUT	CHAR*40	95		
IULINE	INTEGER*2	190		
NDATE	INTEGER*2	136		
NTIME	INTEGER*2	142		

Name	Type	Size	Class
LOGINI			PROGRAM
QBORD			SUBROUTINE
QCLEAR			SUBROUTINE
QCMOV			SUBROUTINE
QCSIZ			SUBROUTINE
QDATE			SUBROUTINE
QTIME			SUBROUTINE

Pass One No Errors Detected
91 Source Lines

APPENDIX E

PROGRAM ROTOR 2

PROGRAM LISTINGS

```
$STORAGE:2
      SUBROUTINE CMASS(PDATA1,CM,SQRTM,SQRTMI)
C     COMPUTES [M] TRACE WITH ALTERNATE M, I ELEMENTS.
C     COMPUTES M**1/2 (SQRTM) AND INVERSE (SQRTMI).
      DIMENSION CM(16),SQRTM(16),SQRTMI(16),PDATA1(20)
      EL=PDATA1(5)/6.
      G=386.
      CM(1)=PDATA1(10)/6./G
      CM(3)=CM(1)*2.
      CM(5)=CM(3)
      CM(7)=CM(1)
      CM(2)=CM(1)*EL**2/3.
      CM(4)=CM(2)*2.
      CM(6)=CM(4)
      CM(8)=CM(2)
      CM(9)=PDATA1(15)/6./G
      CM(11)=CM(9)*2.+PDATA1(6)/G
      CM(13)=CM(9)*2.
      CM(15)=CM(9)+PDATA1(8)/G
      CM(10)=CM(9)*EL**2/3.
      CM(12)=CM(10)*2.+PDATA1(6)/G*PDATA1(7)**2/2.
      CM(14)=CM(10)*2.
      CM(16)=CM(10)+PDATA1(8)/G*PDATA1(9)**2/2.
C     WRITE(6,100)
C 100 FORMAT('OPLANAR MASS TRACE')
C     WRITE(6,110) (CM(I),I=1,16)
C 110 FORMAT(1P6E12.4)
      DO 10 I=1,16
      SQRTM(I)=SQRT(CM(I))
   10 SQRTMI(I)=1./SQRTM(I)
      END
```

```
$STORAGE:2
      SUBROUTINE JACOBI(CKQ8,T,EIGEN)
C     EIGENVALUES AND EIGENVECTORS BY THE JACOBI METHOD.
C
      IMPLICIT REAL*8 (A-H, O-Z)
      DIMENSION A(16,16), T(16,16), AIK(16), EIGEN(16)
     1 ,CKQ8(16,16)
      WRITE(*,205)
    1 DO 2 I=1,16
      DO 2 J=1,16
      T(1,J)=0.0
    2 A(I,J)=CKQ8(I,J)
      N=16
      ITMAX=50
      EPS1=1.D-12
      EPS2=1.D-12
      EPS3=1.D-6
      NM1=N-1
      SIGMA1=0.0
      OFFDSQ=0.0
      DO 5 I=1,N
      SIGMA1=SIGMA1+A(I,I)**2
      T(I,I)=1.D+0
      IP1=1+I
      IF(I.GE.N) GOTO 6
      DO 5 J=IP1,N
    5 OFFDSQ=OFFDSQ+A(I,J)**2
    6 S=2.*OFFDSQ+SIGMA1
      DO 26 ITER=1,ITMAX
      DO 20 I=1,NM1
      IP1=I+1
      DO 20 J=IP1,N
      Q=DABS(A(I,I)-A(J,J))
      IF(Q.LE.EPS1) GOTO 9
      IF(DABS(A(I,J)).LE.EPS2) GOTO 20
      P=2.D+0*A(I,J)*Q/(A(I,I)-A(J,J))
      SPQ=DSQRT(P*P+Q*Q)
      CSA=DSQRT((1.D+0+Q/SPQ)/2.D+0)
      SNA=P/(2.D+0*CSA*SPQ)
      GOTO 10
    9 CSA=1.D+0/DSQRT(2.D+0)
      SNA=CSA
```

```
10    CONTINUE
      DO 11 K=1,N
      HOLDKI=T(K,I)
      T(K,I)=HOLDKI*CSA+T(K,J)*SNA
11    T(K,J)=HOLDKI*SNA-T(K,J)*CSA
      DO 16 K=I,N
      IF(K.GT.J) GOTO 15
      AIK(K)=A(I,K)
      A(I,K)= CSA*AIK(K)+SNA*A(K,J)
      IF(K.NE.J) GOTO 14
      A(J,K)=SNA*AIK(K)-CSA*A(J,K)
14    GOTO 16
15    HOLDIK=A(I,K)
      A(I,K)=CSA*HOLDIK+SNA*A(J,K)
      A(J,K)=SNA*HOLDIK-CSA*A(J,K)
16    CONTINUE
      AIK(J)= SNA*AIK(I)-CSA*AIK(J)
      DO 19 K=1,J
      IF(K.LE.I) GOTO 18
      A(K,J)=SNA*AIK(K)-CSA*A(K,J)
      GOTO 19
18    HOLDKI=A(K,I)
      A(K,I)=CSA*HOLDKI+SNA*A(K,J)
      A(K,J)=SNA*HOLDKI-CSA*A(K,J)
19    CONTINUE
20    A(I,J)=0.0
      SIGMA2=0.0
      DO 21 I=1,N
      EIGEN(I)=A(I,I)
21    SIGMA2=SIGMA2+EIGEN(I)**2
      IF(1.D+0-SIGMA1/SIGMA2.GE.EPS3) GOTO 25
      WRITE(*,204) ITER,SIGMA2,S
      RETURN
25    WRITE(*,202) ITER,SIGMA1,SIGMA2
26    SIGMA1=SIGMA2
      WRITE(*,203) ITER,S,SIGMA1,SIGMA2
      RETURN
201   FORMAT(1P5D14.5)
202   FORMAT(' ITER=',I4,' SIGMA1=',1P1D10.2,' SIGMA2=',1P1D10.2)
203   FORMAT(' NO CONVERGENCE WITH',I4,' ITERATIONS'/' S=',1P1D10.2,
     1 ' SIGMA1=',1P1D10.2,' SIGMA2=',1P1D10.2)
204   FORMAT(' CONVERGENCE OBTAINED WITH',I4,' ITERATIONS'/' SIGMA2='
     1 ,1P1D10.2,' S=',1P1D10.2)
205   FORMAT('OEIGENVALUE ITERATION LOOP')
      END
```

```
$STORAGE:2
      SUBROUTINE CKNORM(CK,SQRTMI,CKQ)
      REAL*8 CKQ(16,16)
      REAL*4 CK(16,16),SQRTMI(16)
C     GENERATE [CK][CM]=CKP
      DO 10 I=1,16
      DO 10 J=1,16
   10 CKQ(I,J)=CK(I,J)*SQRTMI(J)*SQRTMI(I)
C     WRITE(6,100)
C 100 FORMAT('ONORMALIZED STIFFNESS MATRIX, LOWER LEFT TRIANGLE')
C     DO 20 I=1,16
C  20 WRITE(6,110) I,(CKQ(I,J),J=1,I)
C 110 FORMAT(' ROW',I3/(1P6E12.4))
      END
```

```
$STORAGE:2
      SUBROUTINE STIFF(PDATA1,CK)
      REAL*4 PDATA1(20),CK(16,16)
      EL=PDATA1(5)/3.
      ALFA=PDATA1(3)*PDATA1(4)/EL**3
      BETA=PDATA1(1)*PDATA1(2)/EL**3
C     CASING TRIANGULAR STIFFNESS MATRIX
      CK(1,1)=12.*ALFA+PDATA1(11)+PDATA1(13)
      CK(2,2)=4.*EL**2*ALFA
      CK(3,3)=24.*ALFA+PDATA1(16)
      CK(4,4)=2.*CK(2,2)
      CK(5,5)=24.*ALFA+PDATA1(14)+PDATA1(17)
      CK(6,6)=CK(4,4)
      CK(7,7)=12.*ALFA+PDATA1(12)
      CK(8,8)=CK(2,2)
      CK(2,1)=6.*EL*ALFA
      CK(4,1)=CK(2,1)
      CK(3,2)=-CK(2,1)
      CK(6,3)=CK(2,1)
      CK(5,4)=-CK(2,1)
      CK(8,5)=CK(2,1)
      CK(7,6)=-CK(2,1)
      CK(8,7)=-CK(2,1)
      CK(3,1)=-12.*ALFA
      CK(5,3)=CK(3,1)
      CK(7,5)=CK(3,1)
      CK(9,1)=-PDATA1(13)
      CK(4,2)=CK(2,2)/2.
      CK(6,4)=CK(4,2)
      CK(8,6)=CK(4,2)
      CK(13,5)=-PDATA1(14)
C     ROTOR TRIANGULAR STIFFNESS MATRIX
      CK(9,9)=12.*BETA+PDATA1(13)
      CK(10,10)=4.*EL**2*BETA
      CK(11,11)=24.*BETA
      CK(12,12)=2.*CK(10,10)
      CK(13,13)=CK(11,11)+PDATA1(14)
      CK(14,14)=CK(12,12)
      CK(15,15)=12.*BETA
      CK(16,16)=CK(10,10)
      CK(10,9)=6.*EL*BETA
      CK(12,9)=CK(10,9)
      CK(11,10)=-CK(10,9)
      CK(14,11)=CK(10,9)
      CK(13,12)=-CK(10,9)
      CK(16,13)=CK(10,9)
      CK(15,14)=-CK(10,9)
      CK(16,15)=-CK(10,9)
      CK(11,9)=-12.*BETA
      CK(13,11)=CK(11,9)
      CK(15,13)=CK(11,9)
      CK(12,10)=CK(10,10)/2.
      CK(14,12)=CK(12,10)
      CK(16,14)=CK(12,10)
C     FILL IN SYMMETRICAL MATRICIES
      DO 10 I=1,15
      DO 10 J=1,I
   10 CK(J,I+1)=CK(I+1,J)
C     WRITE(6,100)
C 100 FORMAT('OPLANAR STIFFNESS MATRIX, LOWER LEFT TRIANGLE')
C     DO 20 I=1,16
C  20 WRITE(6,110) I,(CK(I,J),J=1,I)
C 110 FORMAT(' ROW',I3/(1P6E12.4))
      END
```

```
$STORAGE:2
      SUBROUTINE SORT(EIGEN,SHAPE,CM,GM,RPS,CPS,SQRTM)
      REAL*8 EIGEN(16),SHAPE(16,16),DUM
      REAL*4 CM(16),GM(16),RPS(16),CPS(16),SQRTM(16)
C     SORT SMALLEST EIGENVALUES(F**2) AND MODE SHAPES FIRST.
C     COMPUTES FREQ IN RAD/SEC AND CYCLES/SEC
      PI=3.14159
      DO 10 I=1,15
      J1=I+1
      DO 10 J=J1,16
      IF(EIGEN(I).LE.EIGEN(J)) GOTO 10
      DUM=EIGEN(I)
      EIGEN(I)=EIGEN(J)
      EIGEN(J)=DUM
      DO 20 K=1,16
      DUM=SHAPE(I,K)
      SHAPE(I,K)=SHAPE(J,K)
   20 SHAPE(J,K)=DUM
   10 CONTINUE
C     NORMALIZE MODE SHAPES
      DO 40 I=1,16
      GM(I)=0.
      RPS(I)=DSQRT(EIGEN(I))
      CPS(I)=RPS(I)/2./PI
      DO 25 J=1,16
   25 SHAPE(J,I)=SHAPE(J,I)*SQRTM(J)
      DUM=DABS(SHAPE(1,I))
      JMAX=1
      DO 30 J=3,15,2
      IF(DUM.GE.DABS(SHAPE(J,I))) GOTO 30
      JMAX=J
      DUM=DABS(SHAPE(JMAX,I))
   30 CONTINUE
      DO 40 J=1,16
      SHAPE(J,I)=SHAPE(J,I)/SHAPE(JMAX,I)
      GM(I)=CM(I)*SHAPE(J,I)**2+GM(I)
   40 CONTINUE
      END
```

```
$STORAGE:2
      PROGRAM ROTOR2
      REAL*8 EIGEN(16),SHAPE(16,16),CK08(16,16)
      REAL*4 CK(16,16),CM(16),SQRTMI(16),SQRTM(16),CKP(16,16),CPS(16)
     1 ,GMASS(16),RPS(16)
      CHARACTER*28 W1(20),W2(20)
      DIMENSION PDATA1(20),PDATA2(20)
      W1(1)='ROTOR SECTION MODULUS, I       '
      W1(2)='ROTOR ELASTIC MODULUS, E       '
      W1(3)='CASE SECTION MODULUS,I         '
      W1(4)='CASE ELASTIC MODULUS,E         '
      W1(5)='ROTOR LENGTH, L                '
      W1(6)='IMPELLER WEIGHT, W             '
      W1(7)='IMPELLER RADIUS OF GYRATION    '
      W1(8)='TURBINE WEIGHT                 '
      W1(9)='TURBINE RADIUS OF GYRATION     '
      W1(10)='CASING WEIGHT                  '
      W1(11)='CASING SPRING TO GROUND K1     '
      W1(12)='CASING SPRING TO GROUND K4     '
      W1(13)='ROTOR BEARING SPRING K51       '
      W1(14)='ROTOR BEARING SPRING K73       '
      W1(15)='ROTOR SHAFT WEIGHT             '
      W1(16)='CASING SPRING TO GROUND K2     '
      W1(17)='CASING SPRING TO GROUND K3     '
      W2(1)='REF. RPM FOR COUPLING COEF.    '
      W2(2)='DIRECT STIFFNESS AT IMPELLER'
      W2(3)='DIRECT DAMPING AT IMPELLER     '
      W2(4)='DIRECT STIFFNESS AT TURBINE    '
      W2(5)='DIRECT DAMPING AT TURBINE      '
      W2(6)='ALFORD COEF AT IMPELLER        '
      W2(7)='ALFORD COEF AT TURBINE         '
      W2(8)='LOWEST RPM OF INTEREST         '
      W2(9)='HIGHEST RPM OF INTEREST        '
      W2(10)='RPM INCREMENT                  '
      W2(11)='MODAL DAMPING, % OF CRITICAL'
      PI=3.14159
      G=386.
      OPEN(5,FILE='CON')
      OPEN(6,FILE='PRN')
      OPEN(7,FILE='PDATA1.STR',STATUS='OLD')
      OPEN(8,FILE='PDATA2.STR',STATUS='OLD')
      WRITE(6,'(A)')'1SIMPLIFIED OVERHUNG ROTOR MODEL.*NO ZERO COEF*'
C     REVIEW AND UPDATE SPRING-MASS-GEOMETRY COEF
    1 READ(7,105) (PDATA1(I),I=1,20)
      REWIND 7
      WRITE(*,110)
  110 FORMAT('0UPDATING INSTRUCTIONS'/
     1 ' <RETURN> ACCEPTS CURRENT VALUE'/
     2 ' ''NEW VALUE''<RETURN> UPDATES DATA VALUE')
    5 WRITE(*,100)
      DO 20 I=1,17
    9 WRITE(*,115)I,W1(I),PDATA1(I)
```

```
  100   FORMAT('OSPRING-MASS INPUT DATA STRING')
        READ(*,130) X
        IF(X.EQ.0) GOTO 20
        PDATA1(I)=X
   20   CONTINUE
        WRITE(7,105) (PDATA1(I),I=1,20)
        REWIND 7
        WRITE(6,100)
        DO 29 I=1,17
   29   WRITE(6,135) I,W1(I),PDATA1(I)
        CALL STIFF(PDATA1,CK)
        CALL CMASS(PDATA1,CM,SQRTM,SQRTMI)
        CALL CKNORM(CK,SQRTMI,CKQ8)
        CALL JACOBI(CKQ8,SHAPE,EIGEN)
        CALL SORT(EIGEN,SHAPE,CM,GMASS,RPS,CPS,SQRTM)
        WRITE(6,140)
  140   FORMAT('OZERO SPEED SYSTEM CHARACTERISTICS'/
     1  ' PLANAR STIFFNESS. LOWER LEFT TRIANGLE')
        DO 10 I=1,16
   10   WRITE(6,145) I,(CK(I,J),J=1,I)
        WRITE(6,141)
  141   FORMAT('OTRACE OF MASS MATRIX')
        WRITE(6,105) (CM(I),I=1,16)
        WRITE(6,142)
        WRITE(5,142)
  142   FORMAT('OMODE NUMBER,FREQ(HZ),GENERALIZED MASS')
        WRITE(6,162)(I,CPS(I),GMASS(I),I=1,16)
        WRITE(*,'(A\)') 'OINPUT MAXIMUM MODE NUMBER OF INTEREST.  '
        READ(*,125) MAXMOD
        WRITE(6,126)
  126   FORMAT('OMODE SHAPES ((CASE,ROTOR) TRANSLATION ,ROTOTION)')
        DO 25 I=1,MAXMOD
        WRITE(6,128) I
  128   FORMAT(1X,' MODE=',I3)
        WRITE(6,167) (SHAPE(J,I),J=1,15,2)
   25   WRITE(6,167) (SHAPE(J,I),J=2,16,2)
   40   READ(8,105) (PDATA2(I),I=1,14)
        REWIND 8
        WRITE(*,150)
  150   FORMAT('OCOUPLING COEF DATA STRING')
        DO 70 I=1,7
   59   WRITE(*,115) I,W2(I),PDATA2(I)
        READ(*,130) X
        IF(X.EQ.0) GOTO 70
        PDATA2(I)=X
   70   CONTINUE
        WRITE(8,105) (PDATA2(I),I=1,14)
        REWIND 8
        WRITE(6,150)
        DO 79 I=1,7
   79   WRITE(6,135) I,W2(I),PDATA2(I)
   50   WRITE(*,170)
  170   FORMAT('ORPM RANGE PARAMETERS AND MODAL DAMPING')
        DO 80 I=8,11
        WRITE(*,115) I,W2(I),PDATA2(I)
        READ(*,130) X
        IF(X.EQ.0)GOTO 80
        PDATA2(I)=X
   80   CONTINUE
   90   WRITE(6,170)
        DO 89 I=8,11
   89   WRITE(6,135) I,W2(I),PDATA2(I)
        WRITE(8,105) (PDATA2(I),I=1,14)
        REWIND 8
C       START LOOP FOR SPEED VARIATIONS
        RPMNOW=PDATA2(8)-PDATA2(10)
```

```
 200    RPMNOW=PDATA2(10)+RPMNOW
        CPSNOW=RPMNOW/60.
        OMEGA=2.*PI*CPSNOW
        WRITE(6,160) RPMNOW,CPSNOW,OMEGA
 160    FORMAT('OSHAFT SPEED'/'   REV/MIN=,'1P1E13.6,
      1 '  REV/SEC=',1P1E13.6,'  RAD/SEC',1P1E13.6/
      2 'OMODE',5X,'REAL',9X,'IMAGINARY',4X,'FREQUENCY(HZ)',4X,
      3 'DAMPING,%')
C       COMPUTE MODES WITH SPEED DEPENDANT STIFFNESS
        CK1=PDATA2(2)*(RPMNOW/PDATA2(1))**2
        CK2=PDATA2(4)*(RPMNOW/PDATA2(1))**2
        DO 30 I=1,16
        DO 30 J=1,16
  30    CKP(J,I)=CK(J,I)
        CKP(3,3)=CKP(3,3)+CK1
        CKP(7,7)=CKP(7,7)+CK2
        CKP(11,11)=CKP(11,11)+CK1
        CKP(15,15)=CKP(15,15)+CK2
        CALL CKNORM(CKP,SQRTMI,CKQ8)
        CALL JACOBI(CKQ8,SHAPE,EIGEN)
        CALL SORT(EIGEN,SHAPE,CM,GMASS,RPS,CPS,SQRTM)
C       COMPUTE DAMPING AND CROSS COUPLING TERMS
C       QUADRATIC FORM IS "S**2+(BR-BIi)*S+(CR-CIi)=0
        DO 210 I=1,MAXMOD
        DUM=(PDATA2(3)*(SHAPE(3,I)-SHAPE(11,I))**2+PDATA2(5)*(SHAPE(7,I
      1 )-SHAPE(15,I))**2)/GMASS(I)*RPMNOW/PDATA2(1)
        BR=PDATA2(11)*RPS(I)/50.+DUM
        BI=(PDATA1(6)*PDATA1(7)**2*SHAPE(12,I)**2+PDATA1(8)*PDATA1(9)**2
      1 *SHAPE(16,I)**2)*OMEGA/GMASS(I)/G
        CR=RPS(I)**2
        CI=DUM*OMEGA/2.
C       COMPUTE ROOTS OF QUADRATIC
        ARGR=(BR**2-BI**2)/4.-CR
        ARGI=CI-BR*BI/2.
        AMAG=SQRT(ARGR**2+ARGI**2)
        AMAG=SQRT(AMAG)
        PHI=ATAN2(ARGI,ARGR)/2.
        DR=AMAG*COS(PHI)
        DI=AMAG*SIN(PHI)
        SR1=DR-BR/2.
        SI1=DI+BI/2.
        SR2=-DR-BR/2.
        SI2=BI/2.-DI
        FREQ1=SIGN(SQRT(SR1**2+SI1**2),SI1)
        FREQ2=SIGN(SQRT(SR2**2+SI2**2),SI2)
        ZETA1=-SR1*100./ABS(FREQ1)
        ZETA2=-SR2*100./ABS(FREQ2)
        FREQ1=FREQ1/2./PI
        FREQ2=FREQ2/2./PI
        WRITE(6,155) I,SR1,SI1,FREQ1,ZETA1
        WRITE(6,165) SR2,SI2,FREQ2,ZETA2
 210    CONTINUE
        IF(RPMNOW.LT.PDATA2(9))GOTO 200
        WRITE(*,180)
 180    FORMAT(1X/' INSERT KEY TO REENTER PROGRAM AS FOLLOWS'/
      1 ' 1 RPM RANGE AND MODE NUMBER'/' 2 COUPLING COEF+RPM RANGE ETC'
      2 /' 3 MASSES, SPRINGRATES, ETC.'/' 4 QUIT PROGRAM',2X,\)
        READ(*,125) NSTART
        GOTO(50,40,1,85) NSTART
 105    FORMAT(1P5E14.6)
 115    FORMAT(1X,I4,A30,1P1E14.6,2X\)
 125    FORMAT(BN,I4)
 130    FORMAT(BN,5F12.0)
 135    FORMAT(1X,I4,A31,1P1E14.6)
 145    FORMAT(' ROW',I3/(1P5E14.5))
 155    FORMAT('0',I3,1P4E14.5)
 165    FORMAT(4X,1P5E14.5)
 162    FORMAT(1X,I3,1P2E14.5)
 167    FORMAT(1X,1P4E14.5)
  85    END
```

SAMPLE RUN

```
SIMPLIFIED OVERHUNG ROTOR MODEL.*NO ZERO COEF*

SPRING-MASS INPUT DATA STRING
   1   ROTOR SECTION MODULUS, I        1.000000E+01
   2   ROTOR ELASTIC MODULUS, E        3.000000E+07
   3   CASE SECTION MODULUS,I          2.000000E+02
   4   CASE ELASTIC MODULUS,E          3.000000E+07
   5   ROTOR LENGTH, L                 2.400000E+01
   6   IMPELLER WEIGHT, W              1.500000E+01
   7   IMPELLER RADIUS OF GYRATION     2.000000E+00
   8   TURBINE WEIGHT                  2.000000E+01
   9   TURBINE RADIUS OF GYRATION      2.000000E+00
  10   CASING WEIGHT                   2.500000E+02
  11   CASING SPRING TO GROUND K1      1.000000E+06
  12   CASING SPRING TO GROUND K4      1.000000E+06
  13   ROTOR BEARING SPRING K51        1.000000E+06
  14   ROTOR BEARING SPRING K73        1.000000E+06
  15   ROTOR SHAFT WEIGHT              2.500000E+01
  16   CASING SPRING TO GROUND K2      1.000000E+06
  17   CASING SPRING TO GROUND K3      1.000000E+06

ZERO SPEED SYSTEM CHARACTERISTICS
PLANAR STIFFNESS. LOWER LEFT TRIANGLE
ROW  1
  1.42625E+08
ROW  2
  5.62500E+08   3.00000E+09
ROW  3
 -1.40625E+08  -5.62500E+08   2.82250E+08
ROW  4
  5.62500E+08   1.50000E+09    .00000E+00   6.00000E+09
ROW  5
   .00000E+00    .00000E+00  -1.40625E+08  -5.62500E+08   2.83250E+08
ROW  6
   .00000E+00    .00000E+00   5.62500E+08   1.50000E+09    .00000E+00
  6.00000E+09
ROW  7
   .00000E+00    .00000E+00    .00000E+00    .00000E+00  -1.40625E+08
 -5.62500E+08   1.41625E+08
ROW  8
   .00000E+00    .00000E+00    .00000E+00    .00000E+00   5.62500E+08
  1.50000E+09  -5.62500E+08   3.00000E+09
ROW  9
 -1.00000E+06    .00000E+00    .00000E+00    .00000E+00    .00000E+00
   .00000E+00    .00000E+00    .00000E+00   8.03125E+06
ROW 10
   .00000E+00    .00000E+00    .00000E+00    .00000E+00    .00000E+00
   .00000E+00    .00000E+00    .00000E+00   2.81250E+07   1.50000E+08
ROW 11
   .00000E+00    .00000E+00    .00000E+00    .00000E+00    .00000E+00
   .00000E+00    .00000E+00    .00000E+00  -7.03125E+06  -2.81250E+07
  1.40625E+07
ROW 12
   .00000E+00    .00000E+00    .00000E+00    .00000E+00    .00000E+00
   .00000E+00    .00000E+00    .00000E+00   2.81250E+07   7.50000E+07
   .00000E+00   3.00000E+08
ROW 13
   .00000E+00    .00000E+00    .00000E+00    .00000E+00  -1.00000E+06
   .00000E+00    .00000E+00    .00000E+00    .00000E+00    .00000E+00
 -7.03125E+06  -2.81250E+07   1.50625E+07
ROW 14
   .00000E+00    .00000E+00    .00000E+00    .00000E+00    .00000E+00
   .00000E+00    .00000E+00    .00000E+00    .00000E+00    .00000E+00
  2.81250E+07   7.50000E+07    .00000E+00   3.00000E+08
```

```
ROW 15
   .00000E+00     .00000E+00     .00000E+00     .00000E+00     .00000E+00
   .00000E+00     .00000E+00     .00000E+00     .00000E+00     .00000E+00
   .00000E+00     .00000E+00   -7.03125E+06   -2.81250E+07    7.03125E+06
ROW 16
   .00000E+00     .00000E+00     .00000E+00     .00000E+00     .00000E+00
   .00000E+00     .00000E+00     .00000E+00     .00000E+00     .00000E+00
   .00000E+00     .00000E+00    2.81250E+07    7.50000E+07   -2.81250E+07
  1.50000E+08

TRACE OF MASS MATRIX
 1.079447E-01  5.757052E-01  2.158895E-01  1.151410E+00  2.158895E-01
 1.151410E+00  1.079447E-01  5.757052E-01  1.079447E-02  5.757052E-02
 6.044905E-02  1.928612E-01  2.158895E-02  1.151410E-01  6.260794E-02
 1.611975E-01

MODE NUMBER,FREQ(HZ),GENERALIZED MASS
   1    2.61863E+02    8.07490E-01
   2    3.65282E+02    2.54921E+00
   3    4.84212E+02    9.62453E-01
   4    7.24182E+02    1.23855E+00
   5    1.61548E+03    8.85749E-01
   6    2.04658E+03    1.75973E+01
   7    2.69500E+03    1.95852E-01
   8    4.58589E+03    4.97898E+00
   9    4.64621E+03    1.95787E-02
  10    6.40026E+03    9.82856E-01
  11    8.12393E+03    2.18926E+00
  12    8.66155E+03    2.79019E-01
  13    9.63382E+03    1.99477E-01
  14    1.16468E+04    4.17392E-01
  15    1.45149E+04    6.52244E-02
  16    1.49164E+04    9.11640E-01
                                                         < CHOOSE MAXIMUM
MODE SHAPES ((CASE,ROTOR) TRANSLATION ,ROTOTION)           MODE OF INTEREST
 MODE=  1
  -2.60334E-01   1.66227E-01   1.00000E+00   3.54022E-02
   3.40498E-02  -1.57145E-02  -1.57603E-03  -1.17430E-02
   2.11694E+00   1.24333E+00   5.92217E-01  -5.43198E-02
  -9.12374E-03  -2.27254E-02  -2.22519E-02  -1.78992E-02
 MODE=  2
   6.46777E-06  -9.23647E-01   1.58477E-04  -2.42736E-04
   4.40293E-02   1.00000E+00   2.68783E-05  -1.08625E-01
   1.51386E+00  -8.09966E-03  -4.58423E-01  -3.71146E-05
  -1.79786E-04   2.57811E-06  -1.69041E-01  -1.74776E-01
 MODE=  3
  -1.93820E-02   1.00000E+00   1.17288E-02  -3.57168E-02
   3.62490E-02   4.77466E-02  -6.58832E-03  -6.48599E-02
   1.81115E+00   1.70266E-01   3.46877E-01   5.59443E-02
  -1.94077E-02  -7.26701E-04  -6.08625E-02   1.08035E-01
 MODE=  4
  -1.68797E-09   4.24507E-08   6.37635E-09   1.32224E-09
  -5.39936E-09   1.00000E+00  -2.26407E-11  -1.02150E-01
   8.15980E-08   7.75198E-09  -4.75846E-08  -1.86178E-09
  -2.25465E-10   1.41076E-10   1.95909E-01   1.63909E-01

COUPLING COEF DATA STRING
   1   REF. RPM FOR COUPLING COEF.     3.000000E+04
   2   DIRECT STIFFNESS AT IMPELLER    1.000000E+04
   3   DIRECT DAMPING AT IMPELLER      1.500000E+01
   4   DIRECT STIFFNESS AT TURBINE     1.000000E+03
   5   DIRECT DAMPING AT TURBINE       1.000000E+01
   6   ALFORD COEF AT IMPELLER         2.000000E+03
   7   ALFORD COEF AT TURBINE          4.000000E+03
```

RPM RANGE PARAMETERS AND MODAL DAMPING

8	LOWEST RPM OF INTEREST	1.000000E+04
9	HIGHEST RPM OF INTEREST	3.000000E+04
10	RPM INCREMENT	1.000000E+04
11	MODAL DAMPING, % OF CRITICAL	1.000000E+00

SHAFT SPEED
REV/MIN=, 1.000000E+04 REV/SEC= 1.666667E+02 RAD/SEC 1.047197E+03

MODE	REAL	IMAGINARY	FREQUENCY(HZ)	DAMPING,%
1	-1.65323E+01	1.64583E+03	2.61956E+02	1.00444E+00
	-1.65988E+01	-1.64564E+03	-2.61926E+02	1.00860E+00
2	-2.57844E+01	2.29734E+03	3.65656E+02	1.12229E+00
	-2.74128E+01	-2.29474E+03	-3.65246E+02	1.19451E+00
3	-3.23904E+01	3.04365E+03	4.84439E+02	1.06414E+00
	-3.31736E+01	-3.04102E+03	-4.84022E+02	1.09081E+00
4	-4.73379E+01	4.55356E+03	7.24762E+02	1.03952E+00
	-4.77563E+01	-4.54885E+03	-7.24013E+02	1.04980E+00

SHAFT SPEED
REV/MIN=, 2.000000E+04 REV/SEC= 3.333333E+02 RAD/SEC 2.094394E+03

MODE	REAL	IMAGINARY	FREQUENCY(HZ)	DAMPING,%
1	-1.65536E+01	1.64738E+03	2.62203E+02	1.00479E+00
	-1.68248E+01	-1.64701E+03	-2.62143E+02	1.02149E+00
2	-2.69885E+01	2.30177E+03	3.66363E+02	1.17243E+00
	-3.35455E+01	-2.29657E+03	-3.65549E+02	1.46052E+00
3	-3.35506E+01	3.04530E+03	4.84704E+02	1.10165E+00
	-3.67350E+01	-3.04003E+03	-4.83872E+02	1.20829E+00
4	-4.87351E+01	4.55977E+03	7.25752E+02	1.06874E+00
	-5.05016E+01	-4.55035E+03	-7.24256E+02	1.10977E+00

SHAFT SPEED
REV/MIN=, 3.000000E+04 REV/SEC= 5.000000E+02 RAD/SEC 3.141590E+03

MODE	REAL	IMAGINARY	FREQUENCY(HZ)	DAMPING,%
1	-1.65156E+01	1.64990E+03	2.62603E+02	1.00096E+00
	-1.71361E+01	-1.64934E+03	-2.62515E+02	1.03891E+00
2	-2.65769E+01	2.30829E+03	3.67400E+02	1.15129E+00
	-4.13355E+01	-2.30049E+03	-3.66193E+02	1.79653E+00
3	-3.39049E+01	3.04717E+03	4.85003E+02	1.11260E+00
	-4.11080E+01	-3.03928E+03	-4.83760E+02	1.35244E+00
4	-4.96955E+01	4.56856E+03	7.27152E+02	1.08771E+00
	-5.37354E+01	-4.55444E+03	-7.24912E+02	1.17976E+00

SELECT ADD'L RUN INFORMATION

RPM RANGE PARAMETERS AND MODAL DAMPING

8	LOWEST RPM OF INTEREST	4.000000E+04
9	HIGHEST RPM OF INTEREST	6.000000E+04
10	RPM INCREMENT	1.000000E+04
11	MODAL DAMPING, % OF CRITICAL	1.000000E+00

SHAFT SPEED
REV/MIN=, 4.000000E+04 REV/SEC= 6.666667E+02 RAD/SEC 4.188787E+03

MODE	REAL	IMAGINARY	FREQUENCY(HZ)	DAMPING,%
1	-1.64155E+01	1.65337E+03	2.63156E+02	9.92800E-01
	-1.75390E+01	-1.65264E+03	-2.63040E+02	1.06121E+00
2	-2.45717E+01	2.31689E+03	3.68765E+02	1.06049E+00
	-5.07607E+01	-2.30649E+03	-3.67178E+02	2.20025E+00
3	-3.34540E+01	3.04930E+03	4.85341E+02	1.09704E+00
	-4.62931E+01	-3.03877E+03	-4.83692E+02	1.52324E+00
4	-5.02235E+01	4.57994E+03	7.28964E+02	1.09653E+00
	-5.74532E+01	-4.56110E+03	-7.25980E+02	1.25953E+00

SHAFT SPEED
REV/MIN=, 5.000000E+04 REV/SEC= 8.333333E+02 RAD/SEC 5.235983E+03

MODE	REAL	IMAGINARY	FREQUENCY(HZ)	DAMPING,%
1	-1.62483E+01	1.65779E+03	2.63859E+02	9.80072E-01
	-1.80413E+01	-1.65689E+03	-2.63718E+02	1.08880E+00
2	-2.10020E+01	2.32750E+03	3.70449E+02	9.02304E-01
	-6.17901E+01	-2.31450E+03	-3.68496E+02	2.66874E+00
3	-3.21986E+01	3.05167E+03	4.85716E+02	1.05505E+00
	-5.22898E+01	-3.03851E+03	-4.83665E+02	1.72065E+00
4	-5.03245E+01	4.59389E+03	7.31185E+02	1.09540E+00
	-6.16497E+01	-4.57035E+03	-7.27461E+02	1.34878E+00

SHAFT SPEED
REV/MIN=, 6.000000E+04 REV/SEC= 1.000000E+03 RAD/SEC 6.283180E+03

MODE	REAL	IMAGINARY	FREQUENCY(HZ)	DAMPING,%
1	-1.60070E+01	1.66314E+03	2.64709E+02	9.62410E-01
	-1.86536E+01	-1.66208E+03	-2.64545E+02	1.12223E+00
2	-1.59050E+01	2.34011E+03	3.72449E+02	6.79656E-01
	-7.43859E+01	-2.32450E+03	-3.70146E+02	3.19844E+00
3	-3.01400E+01	3.05428E+03	4.86127E+02	9.86764E-01
	-5.90773E+01	-3.03847E+03	-4.83680E+02	1.94460E+00
4	-5.00049E+01	4.61043E+03	7.33817E+02	1.08454E+00
	-6.63184E+01	-4.58218E+03	-7.29354E+02	1.44716E+00